Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Luka Saprunov

Avtonomno raziskovanje prostora z mobilno platformo Turtlebot

diplomsko delo

visokošolski strokovni študijski program prve stopnje računalništvo in informatika

Mentor: izr. prof. dr. Danijel Skočaj

Ljubljana 2016

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Mobilni roboti običajno uporabljajo za navigacijo po prostoru zemljevid, na osnovi katerega se lokalizirajo in načrtujejo pot za dosego cilja. Pri izgradnji zemljevida mora robot obiskati dovolj velik del prostora, ki mu omogoča, da zanesljivo zazna ves prostor po katerem naj bi se gibal. Običajno se ta postopek izvaja z ročnim vodenjem robota, obstajajo pa tudi avtomatizirani postopki. Implementirajte postopek za avtonomno raziskovanje prostora in gradnjo zemljevida, ki sloni na zaznavi obrobja raziskanega prostora, in preizkusite nekaj različnih strategij za izbiro raziskovalnih ciljev. Celoten sistem implementirajte in preizkusite na mobilni platformi Turtlebot ter v praksi primerjajte učinkovitost različnih strategij za avtonomno raziskovanje prostora.

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Luka Saprunov, z vpisno številko 63050105 sem avtor diplomskega dela z naslovom:

Avtonomno raziskovanje prostora z mobilno platformo Turtlebot

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom izr. prof. dr. Danijela Skočaja,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 18.marec 2016 Podpis avtorja:

Za nasvete in pomoč pri izdelavi diplomske naloge bi se rad zahvalil mentorju izr. prof. dr. Danijelu Skočaju ter članom Laboratorija za umetne vizualne spoznavne sisteme.

Kazalo

Slike

Tabele

Algoritmi

Seznam uporabljenih kratic

kratica	angleško	slovensko
ROS	Robot Operating System	operacijski sistem za robote
SLAM	Simultanious Localization And Mapping	sočasna lokalizacija in kartiranje
PCA	Primary Component Analysis	analiza glavnih komponent
ICP	Iterative Closest Point	iterativno najbližja točka
SSIM	Structural SIMilarity	strukturna podobnost

Povzetek

Cilj diplomske naloge je implementacija avtonomnega raziskovanja prostora na mobilni platformi Turtlebot, ki uporablja razvojno okolje ROS. Implementirali smo raziskovalni algoritem, ki temelji na zaznavi obrobij in njihovi uporabi kot potencialnih raziskovalnih ciljev. Ker pa lahko algoritem pri izbiri raziskovalnih ciljev upošteva različne kombinacije ocen raziskovanja, smo nato preizkušali in primerjali, katera kombinacija ocen omogoča najbolj učinkovito raziskovanje prostora. Različne strategije izbire raziskovalnih ciljev smo ocenjevali in primerjali na podlagi štirih kriterijev ter referenčne strategije, ki naključno izbira raziskovalne cilje. Podatke za primerjavo smo dobili tako, da smo za vsako strategijo opravili deset uspešnih raziskovanj. Te smo nato kot skupine povprečnih vrednosti in odstopanj primerjali med seboj na podlagi referenčne strategije in kriterijev. Implementirali smo modul, ki poleg raziskanosti prostora in časa raziskovanja omogoča tudi beleženje zgodovine premikov in s tem pot raziskovanja, ki jo opravi Turtlebot.

Ključne besede: avtonomno, raziskovanje prostora, strategija, izbira cilja, ROS, robot, Turtlebot, obrobja.

Abstract

The purpose of this undergraduate thesis is to implement the autonomous exploration of space on the Turtlebot mobile platform that uses the ROS development environment. We implemented an exploration algorithm based on the detection and use of frontier regions as potential exploration goals. Since the algorithm is able to choose an exploration goal based on the combination of different assessments, we tested and compared which combination of goal assessments enables the most efficient exploration of a given space. We assessed and compared different strategies of goal setting on the basis of four criteria and a referential strategy which selects its exploration goals randomly. In order to get the data necessary for comparison, we conducted ten successful explorations per strategy and compared them on the basis of the referential strategy and criteria. We implemented our own module, which tracks the amount of space explored and the time spent for exploration, while also documenting the path traveled by the Turtlebot during exploration.

Keywords: autonomous, space exploration, strategy, goal selection, ROS, robot, Turtlebot, frontiers.

Poglavje 1

Uvod

1.1 Motivacija

Dandanes smo raziskali že večino lažje dostopnih lokacij našega planeta, a še vedno ostaja veliko neraziskanih področij, kot so dna oceanov in morij [4], podzemne jame, deževni gozdovi, tuji planeti [3] ter preostanek vesolja. Teh področij zaradi ekstremnih razmer, kot so radioaktivno sevanje, visok pritisk, pomanjkanje kisika, skrajne temperature, strupi in človeku težko ali v celoti nedostopne lokacije, preprosto ne moremo raziskati sami. Zato uporabljamo robote, ki so sposobni raziskovati okolje in so pogosto vodeni na daljavo. Kadar pa sta komunikacija ali daljinsko vodenje zaradi razmer onemogočena, pa potrebujemo robote, ki so sposobni delovati neodvisno od človeškega nadzora, torej avtonomno. Medtem ko lahko z inteligentnimi sistemi robotov prihranimo veliko časa in sredstev, pa imajo ti (kot npr. roboti v vesolju) na voljo le določene količine energije ali pa sta pot, ki jo lahko prepotujejo, in s tem prostor, ki ga lahko raziščejo, omejena. Zaradi takih omejitev je treba razviti strategije izbiranja raziskovalnih ciljev, ki bodo robotu omogočile, da s čim bolj natančnim, obširnim in hitrim oziroma učinkovitim raziskovanjem danega okolja čim učinkoviteje porabi omejena sredstva. Poleg robotov, ki delujejo v ekstremnih pogojih pa bi dobra avtonomna raziskovalna strategija lahko koristila tudi robotom - hišnim asistentom prikazana na sliki 1.1 in 1.2.

Z njo bi lahko odstranili potrebo po ročnem vodenju robota po prostoru z namenom učenja zemljevida, ki ga potrebuje za izvajanje svojih opravil.

Slika 1.1: Robot kosilnica [27]

Slika 1.2: iRobot Roomba [26]

1.2 Cilj diplomske naloge

Cilj raziskovanja prostora na področju robotike je spoznati okolje, v katerem se robot nahaja, cilj učinkovitega raziskovanja pa je spoznati čim večji del tega okolja, v čim krajšem času in karseda natančno. Poleg implementacije avtonomnega raziskovanja prostora je namen tega diplomskega dela na podlagi teh treh kriterijev analizirati oziroma primerjati učinkovitosti različnih strategij izbire raziskovalnih ciljev. Po predlogu mentorja je bilo delo izvedeno na mobilni platformi Turtlebot z nameščenim okoljem ROS v prostorih Laboratorija za umetne vizualne spoznavne sisteme. Pri raziskovanju je bila uporabljena implementacija starejšega modula ROS-a za raziskovanje prostora [11], čigar pristop temelji na zaznavi in uporabi obrobij za raziskovalne cilje, kakor je to predstavljeno v [6]. Za pravilno delovanje z novejšimi verzijami ROS-a, ki jih je uporabljal Turltebot, smo jo posodobili ter ji odstranili nepotrebne funkcionalnosti. Ker pa je poleg avtonomnega raziskovanja prostora namen tega diplomske dela tudi analiza različnih strategij izbire raziskovalnih ciljev, smo implementirali osnovno opravljanje meritev prej omenjenih kriterijev.

1.3 Zgradba diplomske naloge

V prvem poglavju diplomske naloge sta predstavljena motivacija in cilj diplomske naloge. V drugem poglavju je najprej predstavljena mobilna platforma, na kateri se opravljata raziskovanje in analiza strategij, pri čemer so predstavljene tako komponente strojne kot tudi programske opreme, ki omogoča delovanje in testiranje robota. V tretjem poglavju sledi predstavitev algoritma avtonomnega raziskovanja prostora in natančen opis njegovih komponent, vključno z detekcijo raziskovalnih ciljev ter ključnim postopkom ocenjevanja in izbire raziskovalnih ciljev. V četrtem poglavju je opisana metodologija testiranja in pričakovano delovanje strategij ter analiza in primerjava rezultatov učinkovitosti različnih strategij. V petem poglavju so z zaključnim sklepom predstavljeni rezultati analize strategij izbire raziskovalnih ciljev in problemi pri izdelavi ter možne izboljšave diplomske naloge.

Poglavje 2

Mobilna platforma

Preden lahko razložimo delovanje algoritma za raziskovanje ali metodologije analize in primerjave strategij raziskovanja, je treba predstaviti mobilno platformo Turtlebot, na kateri smo implementirali algoritem in testirali strategije ter z njimi pridobili potrebne podatke za analizo.

2.1 Robot

Enotna opredelitev pojma robot, s katero bi se strinjali vsi, ne obstaja. To je povzel tudi pionir industrijske robotike Joseph Engelberger, ko je dejal, da sicer ne more točno definirati, kaj robot je, a ga prepozna, ko ga vidi [2]. Tako za namen te diplomske naloge robota definiramo kot stroj, sestavljen iz mehanskih in elektronskih enot, ki se jih da sprogramirati. Običajno so roboti načrtovani za hitro in natančno opravljanje del, ki so za človeka bodisi fizično naporna (npr. dvig težkih predmetov), nevarna (npr. ravnanje z nevarnimi snovmi, kot so strupi, kisline, radioaktivni materiali), neprijetna (npr. čiščenje prostorov, sesanje ali košnja trave) ali pa dolgočasna in ponavljajoča (npr. sortiranje pisem in pošiljk na pošti ali delo v proizvodnji za tekočim trakom). Podobno kot živa bitja je tudi robot sestavljen iz nekaterih osnovnih elementov. Eden od teh je računalniški sistem, ki opravlja funkcijo možganov robota. Ta sistem vsebuje navodila za nadzor robota v obliki pro-

gramov in algoritmov, ki običajno obdelujejo vhodne senzorične podatke in pošiljajo navodila aktuatorjem. Prav tako je sestavljen iz senzorjev, ki služijo kot čutila robota. Te naprave ali elementi zaznavajo okolico v obliki fizičnih količin, kot so svetloba, toplota, barva, pritisk, hitrost, dotik, in jih pretvorijo v obliko električnega signala, ki jo računalnik razume. To so, na primer, tipala, kamere, mikrofoni in podobni elementi. Tretji osnovni elementi pa so aktuatorji (angl. actuators) ali efektorji. Služijo kot mišice robota in so mehanizem, prek katerega sistem deluje na okolje oziroma, natančneje, pretvarja prejeti signal v fizično dejanje. To so običajno električni motorji, ki se uporabljajo za premik robota ali robotske roke. Nabor senzorjev in aktuatorjev je odvisen od namena robota in okolja, v katerem robot deluje, ter določa lastnosti robota, kot sta mobilnost in avtonomnost.

2.1.1 Mobilnost

Za razliko od stacionarnih robotov, ki so pritrjeni na tla, kot na primer industrijske robotske roke v tovarnah, so mobilni roboti premični. Glede na okolje, v katerem se gibljejo, lahko mobilne robote nadalje ločimo na kopenske, vodne, zračne in mešane. Zaradi enostavnosti, hitrega učenja uporabe in nizke cene je najbolj priljubljena kopenska različica. Glede na način gibanja lahko te robote razdelimo na robote, ki za premikanje po okolju uporabljajo noge, gosenice ali kolesa. Najbolj zapleteni med naštetimi so roboti z nogami. Prednosti uporabe nog so lažja prehodnost in navigacija po težkem terenu ter bolj naravno in organsko gibanje. Slabosti pa so vedno večja mehanska in elektronska zapletenost ter cena gradnje in večja poraba električne energije. Tem robotom glede na stopnjo kompleksnosti sledijo roboti, ki za premikanje uporabljajo gosenice. Njihova prednost je stalen stik s tlemi, kar preprečuje zdrs robota, boljša razporejenost teže in boljši oprijem. Slabosti pa so možen vpliv na okolje pri obračanju, majhna raznolikosti pri naboru gosenic in omejenost pri nakupu motorjev. Najbolj uporabljani in najlažji način gibanja pa so kolesa. Ta so nizkocenovna, enostavna za načrtovanje in konstrukcijo in nudijo veliko izbiro. Obstajajo različice s osmimi, šestimi,

štirimi, tremi ali dvema kolesoma za gibanje in so enostavni za začetne uporabnike. Kljub temu pa imajo tudi nekatere slabe lastnosti, kot sta zdrs, ki povzroča napačno zaznavanje premika robota, in relativno majhna kontaktna površina s tlemi.

2.1.2 Avtonomnost

Če smo prej robote delili glede na sposobnost premikanja, jih tokrat ločujemo glede na sposobnost samostojnega delovanja. Medtem ko obstaja veliko robotov, ki so odvisni od človeškega nadzora, bodisi neposredno ali na daljavo in delujejo bolj kot naprave na daljinsko vodenje, obstajajo tudi tisti, ki so vodeni s programom. Kljub tej neodvisnosti pa niso pravi avtonomni roboti, saj se ne odzivajo na informacije iz okolja ter zato ponavadi opravljajo le ponavljajoča opravila za katera so bili načrtovani. Za razliko od te vrste robotov pa so avtonomni roboti tisti, ki lahko delujejo neodvisno od nadzora oziroma so sposobni opravljati svoje naloge z visoko stopnjo samostojnosti ter reagirati na dogajanje v okolju. Ta lastnost je zaželena na področjih, kot so raziskovanje vesolja, kjer je stalen nadzor zaradi oddaljenosti in motene komunikacije nemogoč, ali pa pri košnji trave, sesanju in čiščenju tal, kjer je nadzor nezaželen. Da robota štejemo za avtonomnega, mora izpolnjevati tri pogoje:

- programiran mora biti tako, da se na določen način odzove na informacije iz okolja;
- sposoben mora biti pridobiti informacijo o okolici prek senzorjev;
- sposoben mora biti dalj časa delovati brez človeškega poseganja oziroma se v celoti ali delno gibati znotraj delovnega okolja brez človeškega poseganja.

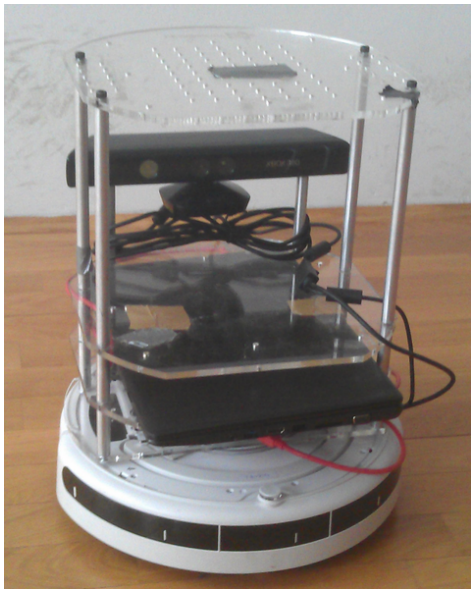
Enostavni primer avtonomnega robota je t. i. robot »zaleti-se-in-pojdi« (angl. bump-and-go robot). Ta se premika v ravni črti, dokler se ne zaleti v oviro, ki jo zazna prek senzorja, se nato obrne v desno, da se izogne najdeni oviri,

in ponovi postopek. Bolj napredni avtonomni roboti so lahko izboljšani z naborom boljših senzorjev, kot so kamere, ki omogočajo stereoskopski vid, ali pa sonarji, s katerimi zaznajo ovire, preden se zaletijo v njih.

2.1.3 Robot želva in Turtlebot

V skupino mobilnih avtonomnih robotov spada tudi t. i. »robot želva« (angl. turtle robot) prikazan na sliki 2.1. To je razred izobraževanih in raziskovalnih robotov, ki so bili zasnovani v poznih štiridesetih letih. Ti roboti oziroma naprave so običajno grajene nizko pri tleh, sposobne so se obračati v zelo majhnih radijih in so krožne oblike ter zato podobne želvi, po kateri so tudi dobile ime. Opremljeni so s senzorji, ki pomagajo pri izogibanju oviram, in sicer s senzorjem za dotik, sonarjem ali kamerami za računalniški vid. Na sliki 2.2 je viden Turtlebot, robot uporabljen pri izdelavi diplomskega dela, ki je ena od modernih verzij robota želve. Gre za nizkocenovni robot, sestavljen iz enostavno dostopnih komponent, medtem ko programsko opremo Turtlebota predstavljata odprtokodni programski sistem in razvojno okolje za robotiko ROS.

Slika 2.1: Robot želva [23]



Slika 2.2: Turtlebot [24]

2.2 Strojna oprema

Enostavno dostopno strojno opremo prej omenjenega Turtlebota sestavljajo iRobot Roomba, dodatni senzor za zajem 3D okolice Kinect in prenosni računalnik, ki opravlja nalogo možganov robota in skrbi za delovanje zapletenih operacij ter obdelavo podatkov.

iRobot Roomba

iRobot Roomba prikazan na sliki 2.3, je član nove generacije avtonomnih robotov, ki opravljajo domača opravila, kot so pomivanje tal, košnja trave in sesanje. Roomba je trenutno svetovno najbolj priljubljen in razširjen robot sesalec. Za avtonomno delovanje sesalec uporablja več strategij, od spiralne poti, sledenja zidu ter zaleti-se-in-pojdi, dokler ne prevozi celotne sobe. Ko so razvijalci Roombe opazili, da robot ni uporaben samo za sesanje, temveč tudi za raziskovalne namene, so izdali še prirejeno verzijo z imenom Create, ki so ji odstranili dele sesalca ter dodali baterijo in vtič za lažjo integracijo računalnika in senzorja Kinect. Ta razred avtonomnih sesalcev robotov je sestavljen iz dveh motorjev za pogon in enega za obračanje, senzorja za dotik, sesalca, kontrolerja ter baterije. Če elemente sesalca zamenjamo z dodatno baterijo, služi iRobot Roomba kot odlična osnova za strojno opremo Turtlebota.

Slika 2.3: iRobot Roomba [26]

Kinect

Medtem ko je Roomba dobra osnova za robota, pa ji je za izgradnjo Turtlebota treba dodati senzor za 3D zajem okolice. Eden od cenovno najbolj ugodnih in dostopnih je Microsoftov Kinect, viden na sliki 2.4. Nabor senzorjev, ki je bil razvit pod imenom projekt Natal, je namenjen rabi za igralno konzolo Xbox. Sama naprava s programsko opremo naj bi bila zmožna prepoznati in slediti gestam, obrazom, gibom in glasovom uporabnikov z namenom bolj naravnega, organskega uporabniškega vmesnika oziroma načina upravljanja konzole. Zaradi nabora senzorjev in ugodne cene pa je postala zelo zanimiva za raziskovalne namene. Naprava je sestavljena iz vga rgb barvne kamera, mikrofona in globinskega senzorja, ki je sestavljen iz infrardečega projektorja in cmos senzorja. Globinski senzor deluje tako, da infrardeči projektor projicira mrežo točk v prostor, ki jih nato zazna s cmos senzorjem. Z uporabo algoritmov računalniškega vida nato izračuna oddaljenosti točk od senzorja.

Slika 2.4: Kinect [21]

Prenosni računalnik

Krmilni računalnik Roombe je sicer dovolj zmogljiv za enostavne procese vodenja sesalca, vendar pa nima dovolj procesorske moči za delovanje razvojnega operacijskega sistema ROS ali za obdelavo podatkov, kot so točkovni oblaki, ki jih zazna 3D senzor Kinect. V ta namen se uporablja prenosni računalnik ASUS x301a, ki z Intelovim dvojedrnim i3 procesorjem nudi dovolj procesorske moči za obdelavo potrebnih podatkov.

2.3 Programska oprema

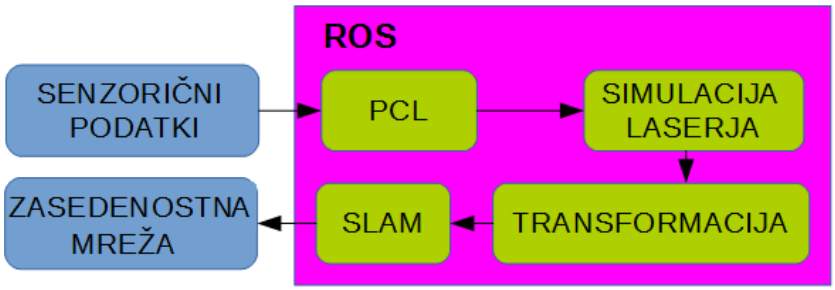
Medtem ko Roomba, Kinect in prenosnik predstavljajo strojno opremo Turtlebota, ta za delovanje še vedno potrebuje tudi programsko opremo. Prenosnik sicer ima nameščen operacijski sistem Ubuntu, vendar to ni dovolj, saj potrebuje ogrodje, ki bo omogočalo nadzor in delovanje strojne opreme ter nudilo osnovo za razvoj aplikacij.

2.3.1 ROS

V ta namen uporabljamo robotski operacijski sistem (angl. Robot Operating System) oziroma skrajšano ROS [9][10]. Gre za skupek programskih ogrodij in algoritmov za razvoj robotskega programa, ki ponujajo funkcionalnost podobni operacijskemu sistemu, oziroma v našem primeru operacijskemu sistemu Ubuntu dodajo potrebne elemente za upravljanje robota. ROS robotu omogoča standardne storitve operacijskega sistema, kot so abstrakcija strojne opreme, nadzor nizkonivojskih storitev, upravljanje s paketi in pošiljanje sporočil. Določene množice ROS procesov so predstavljene v obliki grafa, v katerem v vozliščih potekajo procesi, ki lahko prejemajo, pošiljajo ali multiplicirajo senzorični tok in sporočila. ROS vključuje tudi sposobnost objavljanja in naročanja na teme in podatkovne tokove, kot so slike, laser, nadzor, zvok in točkovni podatki, multipleksiranje informacij, ustvarjanje in uničevanje vozlišč, sinhrono delovanje, razdeljeno na več procesorjev, jeder ali gruč, ter beleženje strežnih parametrov, vodenje simulacij in vizualizacija podatkov. Poleg tega za lažji razvoj aplikacij, posodabljanje robota in obdelavo podatkov uporablja programske knjižice. V našem primeru ROS uporabljamo v treh sklopih; za pridobivanje posodobljenega zemljevida okolja, za navigiranje do zastavljenega cilja ter za iskanje raziskovalnega cilja.

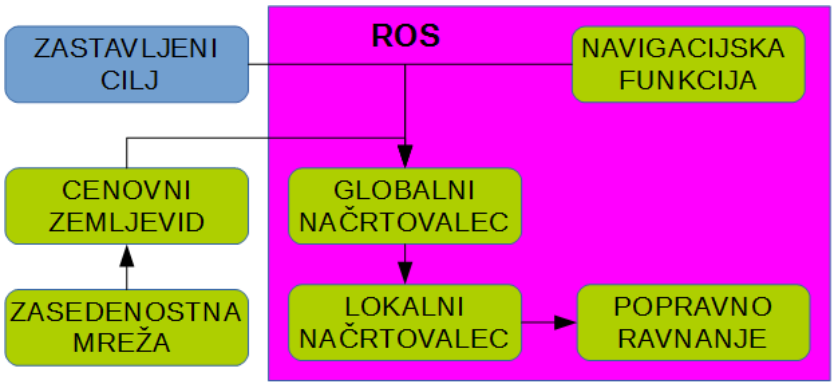
Kot je razvidno z diagrama slike 2.5, v prvem sklopu začnemo s pridobivanjem podatkov o okolici zaznanih s sezorjem Kinect, ki jih nato s knjižico *PCL* filtriramo, grupiramo in drugače obdelamo. Na podlagi teh podatkov se nato simulira zaznavanje laserskega senzorja, katero nato z uporabo trans-

formacijskega paketa pretvorimo iz koordinatnega sistema Kinecta v sistem sveta. Nazadnje na podlagi teh informacij s paketom *gmapping* zgradimo oziroma dopolnimo obstoječo mrežo zasedenostni, ki se jo nato uporabi za iskanje raziskovalnih ciljev z našim raziskovalnim algoritmom.



Slika 2.5: Potek posodabljanja zemljevida

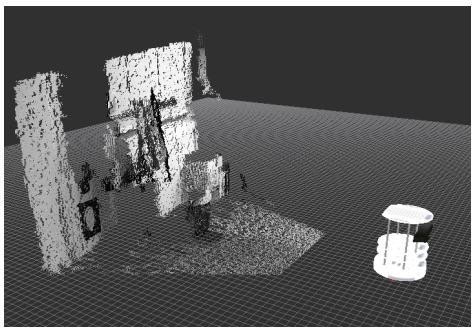
Kot je vidno na sliki 2.6, se drugi sklop paketov ROS ukvarja z iskanjem, načrtovanjem ter navigiranjem poti od trenutnega položaja do zastavljenega cilja, ki ga pridobimo z raziskovalnim algoritmom. Z move base paketom ROS-a poskrbimo za koordinacijo med globalnim in lokalnim načrtovalcem pri načrtovanju in sledenju poti. Ta zato uporabljata cenovni zemljevid ter navigacijsko funkcijo in v primeru problemov vključita popravno ravnanje.



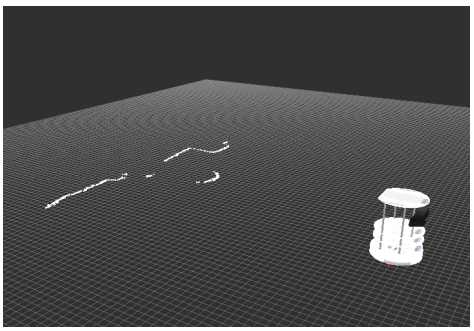
Slika 2.6: Potek navigiranja do cilja

2.3.2 *PCL*

Prva odprtokodna knjižica, ki jo uporabljamo je *PCL*, ki skrbi za obdelavo točkovnih podatkov. ROS-u doda zmogljivost filtriranja, interpolacije in izvajanja podobnih operacij nad točkovnimi podatki, ki jih dobi od 3D senzorja Kinect, kot je prikazano na sliki 2.7. Na sliki 2.8 pa je prikazana simulacija laserskega senzorja iz točkovnih oblakov, ki jo opravlja ROS-ov podpaket povezan s *PCL*.



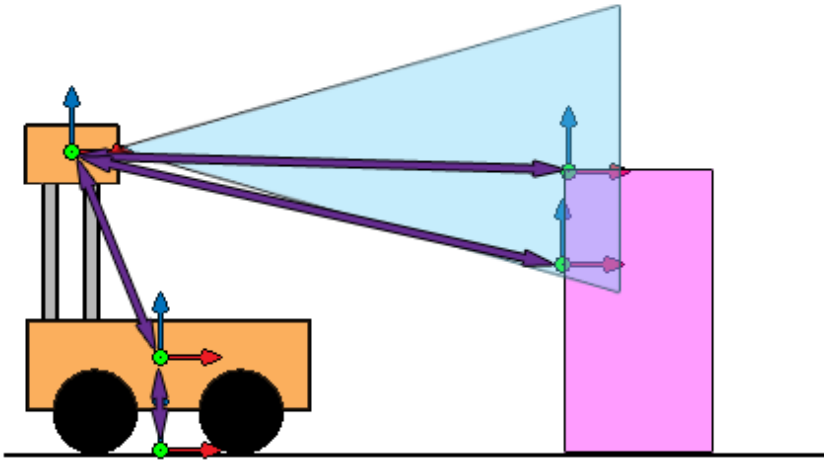
Slika 2.7: *PCL*



Slika 2.8: Simuliran laser

2.3.3 Transformacija

Knjižica transformacij se uporablja za pretvorbo koordinatnih točk iz enega koordinacijski sistem v drugega. V praksi lahko točko, ki jo zazna laser, iz koordinatnega sistema laserja pretvorimo v koordinatni sistem sveta ali pa mobilne osnove robota, ki lahko nato podatke pravilno interpretira pri grajenju zemljevida ali navigaciji robota, kot je to razvidno s slike 2.9. Knjižica za pretvorbe med koordinatnimi sistemi na abstraktni ravni uporablja transformacijsko drevo. To določa zamike tako v translaciji kot tudi v rotaciji med dvema koordinatnima sistemoma. Gradi se v obliki vozlišč s povezavami starš-otrok. Te povezave hranijo razliko med transformacijami, kar omogoča, da z enim prehodom med vozlišči ugotovimo transformacijo med danima koordinatnima sistemoma.



Slika 2.9: Transformacija

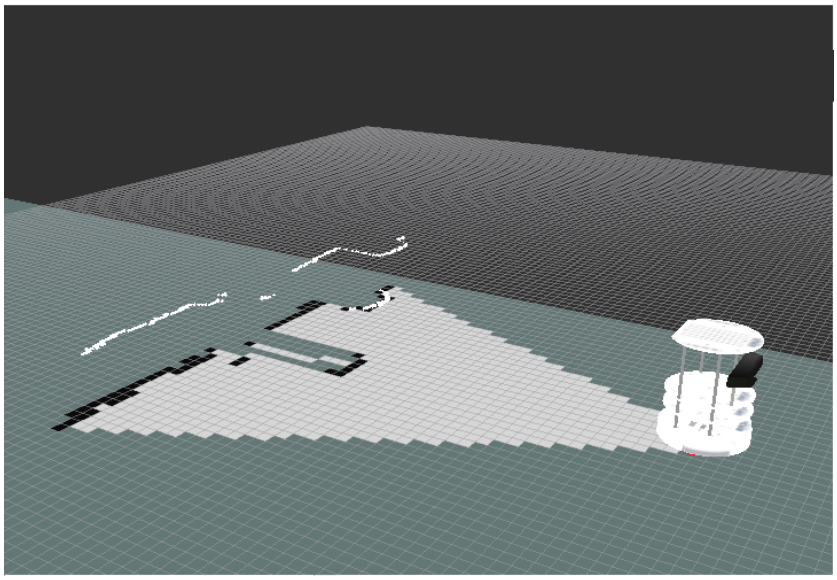
2.3.4 *Gmapping*

Gmapping je ROS paket, ki implementira različico SLAMa [28] fastSlam [29] algoritma, potrebno za gradnjo zemljevida danega okolja. SLAM je kratica za sočasno lokalizacijo in kartiranje (angl. simultaneous localization and mapping). To je algoritem oziroma tehnika, ki jo digitalne naprave uporabljajo za sestavo novega ali posodobitev obstoječega zemljevida v neznanem okolju ter sočasno spremljanje pozicij naprave v obstoječem okolju. Z drugimi besedami, SLAM je proces sestavljanja zemljevida v neznanem okolju, ko se po njem gibljemo in beležimo, kje se nahajamo. Algoritem je sestavljen iz dveh problemov: kartiranja in lokalizacije. Kartiranje je problem integriranja informacij, zbranih z naborom senzorjev, v konsistenten model ter predstavitve teh informacij. Odgovarja na vprašanje, kako zgleda svet. Glavni del kartiranja je predstavitev okolja in interpretiranje senzoričnih podatkov oziroma sestava zemljevida. Zemljevid se uporablja za določanje položaja in orientacije naprave znotraj okolja ter prikaz okolja z namenom planiranja in navigacije. Za dobro gradnjo zemljevida potrebujemo natančno lego oziroma

2D položaj in kot orientacije robota, saj lahko le tako pravilno določimo, na katere pozicije lahko kartiramo novo pridobljene senzorične podatke. Če kartiranje odgovarja na vprašanje, kako zgleda svet, pa lokalizacija odgovarja na vprašanje, kje v svetu se nahajamo. Glavna naloga lokalizacije je ocenjevanje lege, torej položaja in orientacije robota, glede na dani zemljevid in okolico. Običajno je rešitev problema lokalizacije sledenje legi, kje naprava je oziroma, za koliko se je premaknila, pri čemer mora biti znana začetna lega naprave. Za dobro lokalizacijo potrebujemo natančen zemljevid, saj mora robot preko njega in obstoječih značilk – značilnih točk (angl. landmark) iz sveta razbrati, kje se nahaja. Pri uporabi SLAM algoritma lahko pride do problemov zaradi napak pri delovanju robota. Šum senzorjev, zdrs koles in druge napake senzorjev in aktuatorjev povzročajo nepravilnosti pri pridobivanju podatkov, zaradi katerih se pojavijo problemi pri kartiranju, ko se zaznan objekt kartira na napačno lokacijo, ali pri lokalizaciji, ko se robotu glede na okolico določi napačna orientacija ali pozicija. Če se te napake kopičijo, to čez čas občutno popači gradnjo zemljevida in s tem sposobnost določanja dejanske poze naprave z dovolj veliko natančnostjo.

2.3.5 Mreža zasedenosti

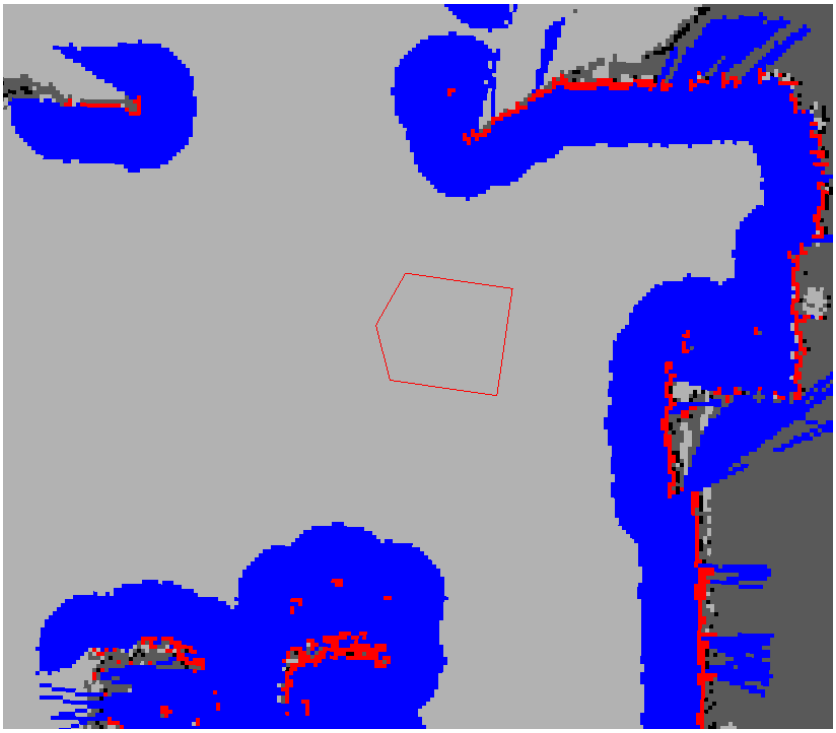
ROS paket za navigacijo uporablja različne predstave prostora, med drugim tudi mrežo zasedenosti (angl. occupancy grid). Gre za enostavno predstavo prostora v obliki 2D kartezijske mreže, ki v celicah hrani verjetnost, da je prostor, ki ga celica predstavlja, zaseden. Če je verjetnost zasedenosti celice manjša od 0,05, ima vrednost –1 oziroma je označena kot »NERAZISKANA«. Če je ta verjetnost manjša od 0,5, ima vrednost 0 in je obarvana kot raziskana in »PROSTA«, sicer pa ima vrednost 100 in se šteje, da je celica raziskana ter »ZASEDENA«, oziroma predstavlja oviro. Mreža oziroma vrednosti njenih celic se posodabljajo na podlagi senzoričnih podatkov robota, ki jih dobimo preko paketa *gmapping*, kot je razivdno iz slike 2.10.



Slika 2.10: Mreža zasedenosti

2.3.6 Cenovni zemljevid

Poleg mreže zasedenosti paket za navigacijo vsebuje cenovni zemljevid (angl. costmap), prikazan na sliki 2.11, ki se uporablja za iskanje in načrtovanje poti ter navigacijo robota po prostoru. Podobno kot mreža zasedenosti, na kateri temelji, je cenovni zemljevid 2D kartezijska mreža. Za razliko od mreže zasedenosti pa v celicah, v obliki vrednosti med 0 in 255, hrani prehodnost dela prostora, ki ga predstavlja. Ko se cenovni zemljevid ustvarja ali posodablja, to počne z napihovalnim radijem (angl. inflation radius) in mrežo zasedenosti. Na podlagi senzoričnih podatkov pridobljenih z *gmapping*-om se najprej posodobi mreža zasedenosti, nato pa se napihnejo vrednosti vseh celic cenovnega zemljevida znotraj napihovalnega radija okoli zasedenih celic mreže zasedenosti. Tako dobljeni cenovni zemljevid se uporablja pri navigaciji robota do cilja z namenom izogibanja oviram. Če središče robota ni v napihnjenem območju oziroma rob robota ni znotraj območja, ki ga zavzema ovira, je robot prost in se lahko premika, v nasprotnem primeru pa se zaradi nevarnosti trka ne more premikati po teh celicah.



Slika 2.11: Cenovni zemljevid [15]

2.3.7 Move base

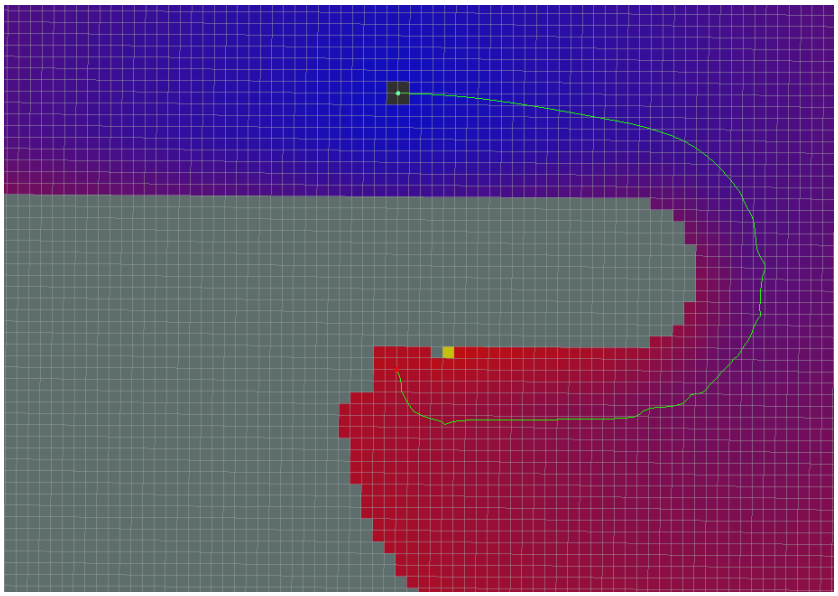
Tako kot *gmapping* je move base podpaket navigacije. Odgovoren je za načrtovanje poti in navigacijo robota po prostoru do zastavljenega cilja. Ko prejme cilj, prevzame nadzor nad robotom in ga poskuša pripeljati do zastavljenega cilja. To doseže s povezovanjem lokalnega in globalnega načrtovalca ter popravnim ravnanjem (angl. recovery behaviour) robota.

2.3.8 Navigacijska funkcija

Navigacijska funkcija, skrajšano *Navfn*, je podpaket, ki za hitro navigacijsko funkcijo implementira Dijkstrov preiskovalni algoritem [**?**]. Ta se nato lahko uporabi za ustvarjanje načrtov za mobilno bazo, da z njo poiščemo najkrajšo pot od začetne do končne pozicije. Podpaket uporabljata tako globalni kot lokalni načrtovalec.

2.3.9 Globalni načrtovalec

Globalni načrtovalec je odgovoren za ustvarjanje visokonivojskega načrta, ki mu sledi robot. Za dani oddaljeni cilj si ustvari površni načrt potovanja, ki mu lokalni načrtovalec nato poskuša slediti. Deluje optimistično, saj predvideva da je robotova stopinja (angl. robot footprint) okrogle oblike in ne upošteva dinamike robota – kako ostre zavoje lahko dela robot ali kako hitro lahko potuje, zaradi česar je lahko načrtovana pot neizvedljiva. Delovanje globalnega načrtovalca lahko vidimo na sliki 2.12.

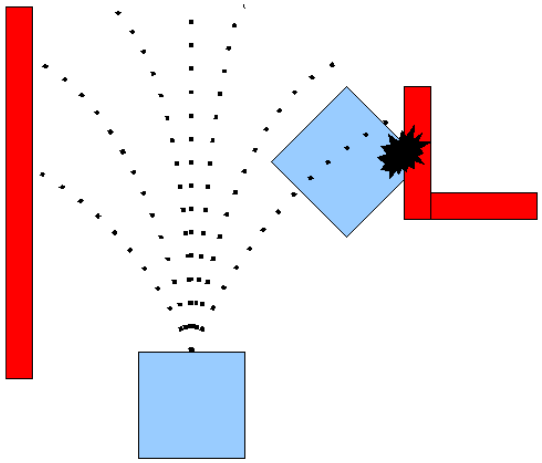


Slika 2.12: Delovanje globalnega načrtovalca [12]

2.3.10 Lokalni načrtovalec

Z uporabo zemljevida lokalni načrtovalec ustvari kinematsko trajektorijo (angl. kinematic trajectory) za robota od začetka do konca zastavljene poti. To naredi tako, da najprej ustvari lokalno funkcijo v obliki mreže, s katero vkodira ceno potovanja skozi celice te mreže. Kontroler nato na podlagi te funkcije oziroma vrednosti teh celic določi linearno in kotno hitrost robota, tako da ju diskretno vzorči. Na ta način za vsako vzorčeno hitrost simulira

prihodnje stanje robota glede na trenutno stanje oziroma določi, kje bi se robot nahajal, če bi nekaj časa uporabljal določeno hitrost, kar je razvidno s slike 2.13. Na koncu oceni vsako dobljeno trajektorijo, pri čemer upošteva bližino cilja in prepreke.



Slika 2.13: Delovanje lokalnega načrtovalca [13]

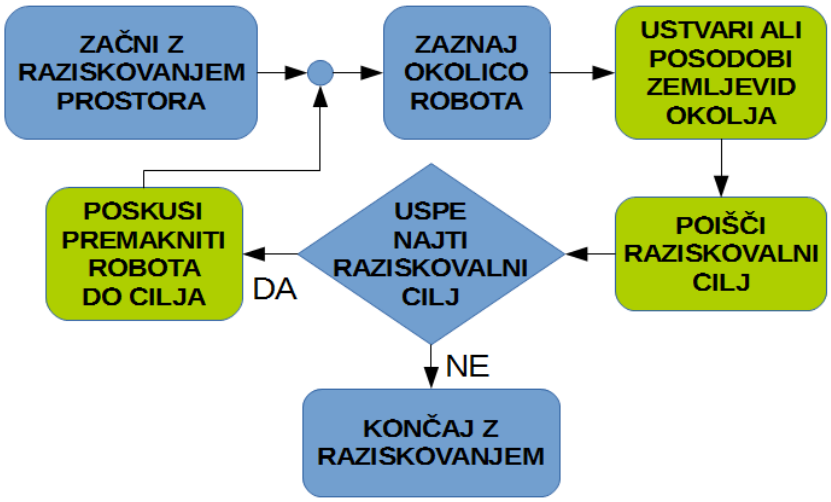
2.3.11 Popravno ravnanje

Paket popravno ravnanje (angl. recovery behaviour) se uporabi, ko robot misli, da se je zataknil, bodisi ker se je zaletel v oviro, ker ne zmore najti poti do danega cilja ali ker misli, da se je zataknil zaradi napačno izdelanega zemljevida. S popravnim ravnanjem si robot poskuša popraviti predstavo sveta, za katero misli, da je napačna. Najprej poskusi na zemljevidu odstraniti ovire iz okolice, nato z obratom na mestu počisti in ponovno zazna okolico. Če še vedno misli, da je zataknjen, bo poskusil z bolj agresivno različico prej omenjenega odstranjevanja ovir z zemljevida in čiščenjem ter ponovno zaznavo okolice z obračanjem na mestu. Če je popravno ravnanje uspešno, robot nadaljuje z navigacijo do cilja. Če pa po izvedbi vseh akcij cilj še vedno ni dosegljiv, se potovanje do izbranega cilja opusti.

Poglavje 3

Postopek raziskovanja

Postopek raziskovanja poteka v več zaporednih sklopih. Začne se z vrtenjem robota okoli lastne osi. Ta namreč zaradi omejenega vidnega polja senzorja Kinect ne vidi celotne okolice in bi zaradi tega lahko pri raziskovanju izpustil potencialni raziskovalni cilj. Zato ga pred vsakim iskanjem novega raziskovalnega cilja, tako na začetku kot med raziskovanjem prostora, zavrtimo okoli lastne osi. S tem poskrbimo, da pravilno zazna celotno vidno okolico. Zaznavi okolice sledi posodobitev zemljevida. Kot smo že omenili v prejšnjem poglavju, robot s senzorjem Kinect poskenira okolico ter na podlagi podatkov, ki jih prej obdela s knjižico *PCL* in nato transformira v pravi koordinatni sistem, s paketom ROS-a *gmapping* doda na zemljevid. Temu sledi iskanje raziskovalnega cilja, ki ga opravimo s prirejenim raziskovalnim ROS paketom. Ta na zemljevidu poišče vse potencialne cilje, jih v skladu z vnaprej določeno strategijo raziskovanja oceni in po tej ceni sortira ter nato izmed njih izbere najboljšega, ki je dosegljiv. V kolikor robotu uspe najti cilj, se ta nato poskusi premakniti do njega z uporabo navigacijskih paketov ROS-a na način, ki smo ga opisali že v prejšnjem poglavju. Ne glede na to ali uspešno doseže cilj, robot nato ponovi postopek raziskovanja. Ko robotu ne uspe najti raziskovalnih ciljev, se celoten postopek raziskovanja zaključi. Delovanje celotnega postopka lahko vidimo na sliki 3.1 ter pseudokodi algoritma 1.



Slika 3.1: Diagram postopka raziskovanja

Algorithm 1 *glavna_zanka*

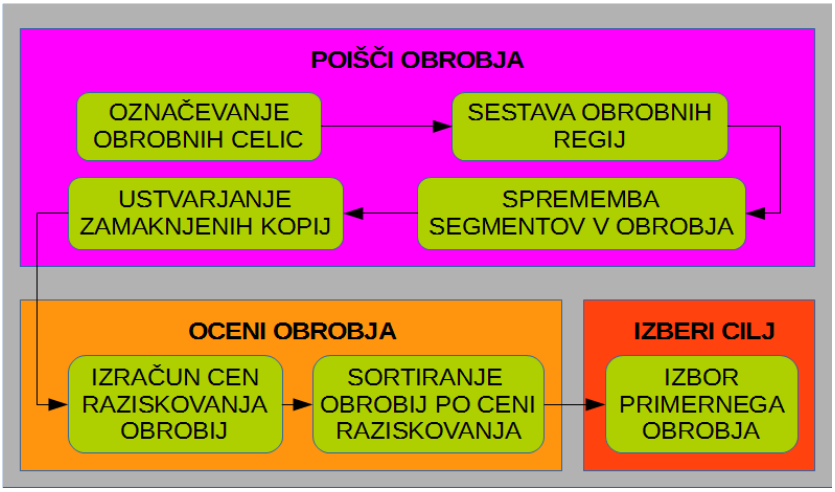
```
1: while končaj ≠ false do
2:     rotiraj_robota()
3:     obrobje o = izberi_raziskovalni_cilj()
4:     if o ≠ null then
5:         premakni_do_cilja(o)
6:     else
7:         končaj = true
8:     end if
9: end while
```

3.1 Pridobivanje raziskovalnih ciljev

Preden lahko poiščemo raziskovalne cilje, je treba opredeliti, kaj ti sploh so. Raziskovanje na področju robotike lahko opredelimo kot grajenje zemljevida med gibanjem skozi neraziskano okolje. Osrednje vprašanje raziskovanja glede na dane podatke o svetu je, kam naj se robot premakne, da pridobi

čim več novih podatkov. Za izbiro cilja uporabljamo prilagojen pristop raziskovanja, ki izhaja iz leta 1997 in temelji na konceptu zaznave in uporabe obrobij, mej med raziskanim in neraziskanim delom okolja za raziskovalne cilje [6]. Razlog za uporabo obrobij kot raziskovalnih ciljev je, da ko se robot premakne do izbranega obrobja, vidi v neraziskani del okolja in s tem novo zaznane podatke doda na zemljevid. Na ta način del neraziskan okolja postane raziskan, zaradi česar pa se premakne meja med raziskanim in neraziskanim. Medtem ko omenjeni pristop razišče okolje in zgradi njegov zemljevid, je naša naloga to opraviti kar se da hitro in pri tem raziskati čim večji del okolja. Eden od dejavnikov, ki vpliva na to, je izbira raziskovalnega cilja. Če odkrijemo več obrobij in s tem več potencialnih raziskovalnih ciljev, potrebujemo strategijo za izbiro raziskovalnega cilja. Prvotni pristop iz leta 1997 uporablja enostavno strategijo, ki za raziskovalni cilj preprosto izbere robotu najbližje obrobje. Medtem ko ta pristop deluje, pa za naše potrebe uporabljamo razširjeno verzijo, ki poleg oddaljenosti od obrobja za izbiro raziskovalnega cilja upošteva še dve dodatni oceni. To sta ocena spremembe v orientaciji med robotom in obrobjem ter ocena količine informacij, ki bi jih pridobili z raziskovanjem danega obrobja. Zaradi večjega števila ocen za izbiro raziskovalnega cilja po eni strani nismo več omejeni le na oceno oddaljenosti, po drugi strani pa zaradi tega potrebujemo način, s katerim lahko določimo, v kolikšni meri upoštevamo določeno oceno pri izračunu cene raziskovanja obrobja. Na ta način določimo strategijo izbire raziskovalnih ciljev in s tem potek raziskovanja. To dosežemo tako, da za ceno raziskovanja uporabimo vsoto uteženih ocen, pri čemer uteži določajo, v kolikšni meri se upošteva posamezna ocena oziroma atribut. Samo pridobivanje raziskovalnega cilja opravimo s funkcijo IZBERI RAZISKOVALNI CILJ prikazano v algortimu 2 in sliki 3.2. Njene namen je, za določeno raziskovalno strategijo, poiskati najbolj primeren dosegljiv raziskovalni cilj. Postopek pridobivanja raziskovalnih ciljev začnemo z uporabo funkcije POIŠČI OBROBJA. Z njo na posodobljeni zasedenostni mreži poiščemo vsa obrobja ter ustvarimo zamaknjene kopije. S funkcijo OCENI OBROBJA uredimo seznam obrobij,

na podlagi cene raziskovanja, ki smo jo izračunali v skladu z izbrano strategijo raziskovanja. Nazadnje s funkcijo IZBERI CILJ z ustvarjenega seznama izberemo najboljši dosegljivi cilj ter ga posredujemo naprej k paketu *move base*, ki poskrbi za navigacijo do cilja.



Slika 3.2: Delovanje algortima raziskovanja

Algorithm 2 *izberi_raziskovalni_cilj*

Input: mreža_zasedenosti

Output: seznam_obrobij

1: mreža_zasedenosti = *dobi_mrežo_zasedenosti()*
2: seznam_obrobij = *poišči_obrobja*(mreža_zasedenosti)
3: *oceni_obrobja*(seznam_obrobij)
4: *izberi_cilj*(seznam_obrobij)
5: **return** seznam_obrobij

3.2 Funkcija POIŠČI OBROBJA

Naloga te funkcije je na dani zasedenostni mreži najti obrobja, območja prostega raziskanega prostora, ki mejijo na neraziskano. Če bi to počeli po

zveznem prostoru, bi navedena opredelitev zadostovala. Ker pa imamo na voljo le diskretno predstavo sveta v obliki mreže zasedenosti, je treba obrobje iskati na ravni posameznih celic, ki jih nato združimo v skupno območje. Za to uporabimo postopke in metode, podobne zaznavi robov in ekstrakciji regij s področja računalniškega vida. Postopek iskanja obrobij poteka v treh korakih. Kot lahko vidimo na pseudokodi algoritma 3 v prvem koraku s funkcijo OZNAČI OBROBNE CELICE na mreži zasedenosti poiščemo in označimo vse obrobne celice. Nato v drugem koraku s funkcijo SESTAVI OBROBNE REGIJE na mreži zasedenosti z označenimi obrobnimi celicami za vsako obrobno celico izračunamo orientacijo proti neraziskanem. Sosednje obrobne celice nato združimo v obrobno regijo. Nazadnje s funkcijo DOBI OBROBJA s seznama obrobnih regij izločimo vse neprimerne regije, na podlagi preostalih pa ustvarimo seznam obrobij.

Algorithm 3 *poišči_obrobja*

Input: mreža_zasedenosti

Output: seznam_obrobij

1: mreža_zasedenosti = *dobi_ mrežo_zasedenosti* ()
2: *označi_obrobne_celice*(mreža_zasedenosti)
3: seznam_regij = *sestavi_obrobne_regije*(mreža_zasedenosti)
4: seznam_obrobij = *dobi_obrobja*(seznam_regij)
5: **return** seznam_obrobij

3.3 Funkcija OZNAČI OBROBNE CELICE

Kot smo že omenili, mreža zasedenosti prikazana na sliki 3.3 za predstavo prostora uporablja tri vrednosti. Zaradi potrebe pri iskanju obrobij smo prvotnim definiranim vrednostim dodali še vrednost 1, s katero celico označimo kot »OBROBNO«. Podobno kot je obrobje meja med raziskanim in neraziskanim, štejemo, da je vsaka raziskana in »PROSTA« celica, ki meji na »NERAZISKANO«, del obrobja, oziroma je obrobna celica. Postopek preverjanja obrobnosti je zelo podoben postopku zaznave robov (angl. edge

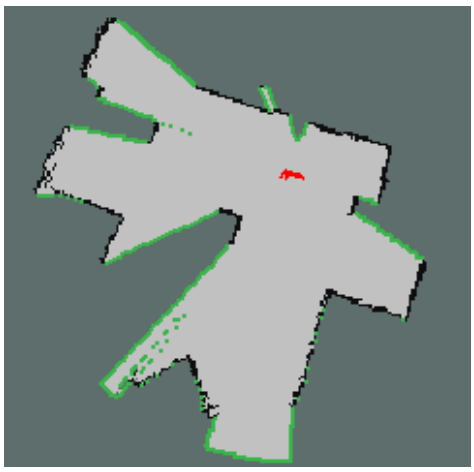
detection) s področja računalniškega vida. Kot je razvidno z algoritma 4, za vsako »PROSTO« celico mreže zasedenosti preverimo, ali v štiri-sosednosti meji na »NERAZISKANO« sosedo. Če izpolnjuje ta pogoj, jo označimo kot »OBROBNO«. Tako označena mreža zasedenosti, vidna na sliki 3.4, je pripravljena za nadaljnje postopke iskanja obrobij.

Algorithm 4 *označi_obrobne_celice*

Input: mreža_zasedenosti
Output: mreža_zasedenosti

```
1: for all celica c na mreži_zasedenosti do
2:     if c == "PROSTA" then
3:         seznam S = 4_sosednosti( c, mreža_zasedenosti )
4:         for all celica t na seznamu S do
5:             if t == ''NERAZISKAN" then
6:                 c = "OBROBNA"
7:             end if
8:         end for
9:     end if
10: end for
11: return mreža_zasedenosti
```

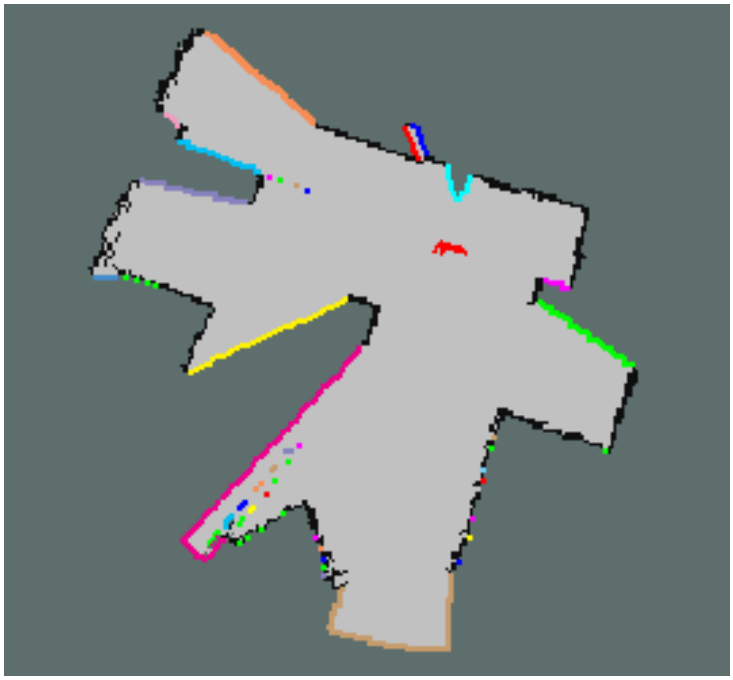
Slika 3.3: Mreža zasedenosti



Slika 3.4: Označene obrobne celice

3.4 Funkcija SESTAVI OBROBNE REGIJE

Glavni del funkcije je združevanje posameznih obrobnih celic v skupne regije s postopkom podobnim barvanju objektov (angl. blob coloring) s področja računalniškega vida, kjer posamezne objekte zaznamo in ločimo od ostalih tako, da jih »pobarvamo« z različnimi barvami, kot je prikazano na sliki 3.5. Poleg preprostega seznama celic obrobne regije, pa za kasnejše ocenjevanje obrobij potrebujemo podatke o njihovi oddaljenosti, orientaciji in velikosti. Ker »OBROBNE« celice same niso sposobne hraniti teh informacij, na podlagi njihovih sosed ter njih samih, na način povzet v algoritmu 5, ustvarimo obrobne točke. To so objekti, ki v obliki vektorjev hranijo položaj in orientacijo dane celice, kot je predstavljeno v tabeli 3.1. Orientacija obrobne točke predstavlja povprečno smer proti »NERAZISKANEMU«, ki jo izračunamo kot normalizirano vsoto vektorjev od celice do njenih »NERAZISKANIH« sosed.



Slika 3.5: Obrobne regije

spremenljivka	namen
`pozicija`	2D vektor, ki hrani x in y koordinate položaja točke
`orientacija`	2D vektor, ki predstavlja orientacijo obrobne točke

Tabela 3.1: Objekt obrobna točka

Algorithm 5 *ustvari_obrobno_točko*

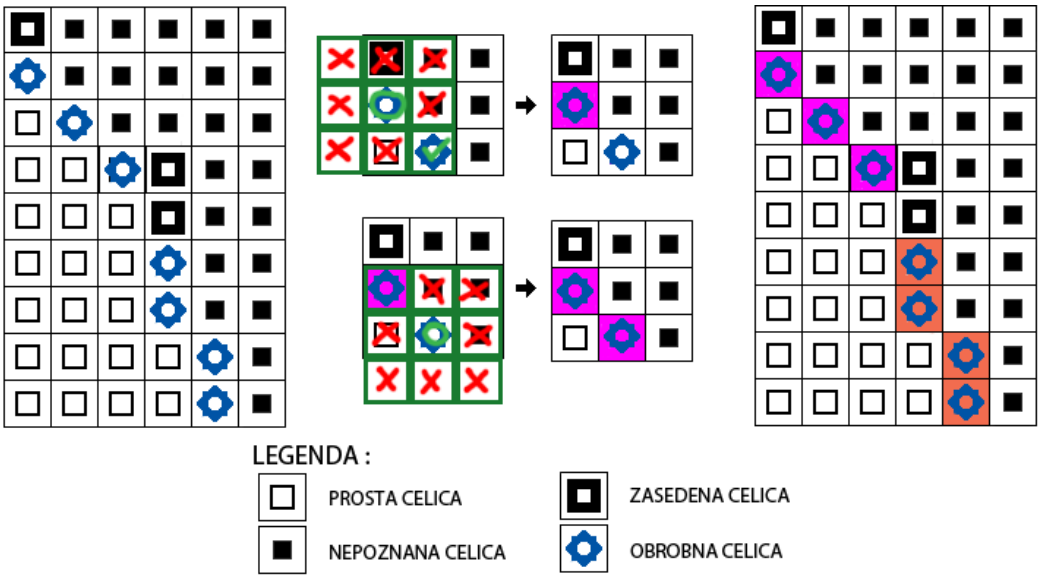
Input: indeks_celice, mreža_zasedenosti

Output: obrobna_točka

```
1: pozicija = nov vektor ( c mod mreža_zasedenosti.širina, c/ mreža_zasedenosti.širina)
2: smer = nov vektor ( 0, 0 )
3: števec = 0
4: for all celica c na seznamu S do
5:     if c == '"NERAZISKAN" then
6:         smer += trenutna_smer
7:         števec = 0
8:     end if
9: end for
10: smer /= števec
11: return nova obrobna_točka(pozicija, smer)
```

Tako ustvarjeno obrobno točko dodamo na seznam, ki predstavlja trenutno obrobno regijo, najdeno »OBROBNO« celico pa na mreži zasedenosti označimo z indeksom trenutne obrobne regije in jo tako ločimo od še neobdelanih obrobnih točk ter s tem preprečimo njeno ponovno obravnavo. Nato za trenutno izbrano celico preverimo, ali je katera izmed njenih osmih sosednjih celic prav tako »OBROBNA«. Za vsako najdeno obrobno sosedo ponavljamo postopek označevanja celic, izdelave obrobne točke in njeno dodajanje k regiji

ter iskanje sosednjih »OBROBNIH« celic. Ko teh zmanjka, smo zaključili s sestavljanjem trenutne obrobne regije in jo dodamo na seznam obrobnih regij. Nato povečamo indeks obrobne regije in dokler ne dosežemo konca mreže zasedenosti, poskušamo najti »OBROBNE« celice drugih obrobnih regij, pri katerih ponovimo postopek sestavljanja regije. Postopek sestavljanja regij je viden na sliki 3.6. Tako dobljen seznam obrobnih regij pozneje uporabimo za izdelavo obrobij. Pseudokodo celotnega procesa si lahko ogledamo v algortimu 6.



Slika 3.6: Sestavljanje regij

Algorithm 6 *sestavi_obrobne_regije*

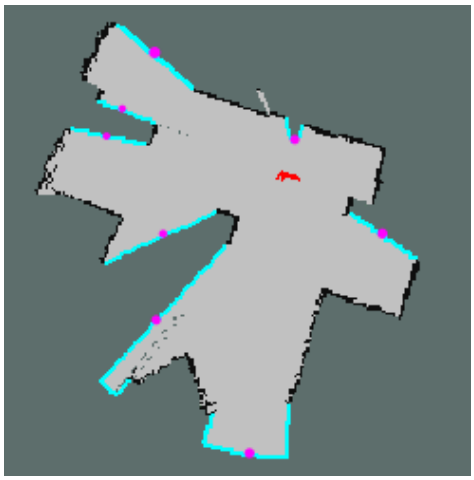
Input: mreža_zasedenosti

Output: seznam_regij

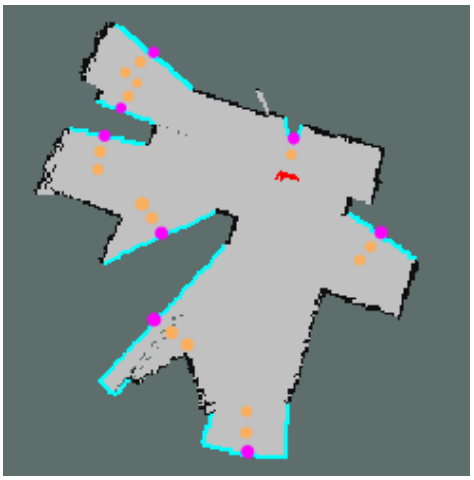
```
1: senam_regij = nov seznam
2: indeks_regij = 1
3: for all celica c na mreži_zasedenosti do
4:     if c == ''OBROBNA" then
5:         seznam_sosed = nov seznam
6:         obrobna_regija = nov seznam
7:         Enqueue(seznam_sosed, c)                ▷ dodaj c na seznam_sosed
8:         while
9:             docelica c_t = Dequeue( seznam_sosed )   ▷ vzemi element s seznama sosed
10:            if c_t == ''OBROBNA" then
11:                obrobna_točka = ustvari_obrobna_točka( c_t, mreža_zasedenosti )
12:                Enqueue( obrobna_regija, obrobna_točka )
13:                seznam = 8_sosednost( c_t, mreža_zasedenosti )
14:                for all c doelice c_ na seznamu
15:                    if c_i == ''NERAZISKAN" then
16:                        Enqueue( seznam_sosed, c_i )
17:                    end if
18:                end for
19:            end if
20:        end while
21:        Enqueue( seznam_regij, obrobna_regija )  ▷ dodaj obrobno_regija na seznam_regij
22:    end if
23: end for
24: return seznam_regij
```

3.5 Funkcija DOBI OBROBJA

Funkcijo DOBI OBROBJA uporabimo za ustvarjanje seznama obrobij na podlagi seznama obrobnih regij. S tega seznama najprej izločimo vse regije, ki so premajhne, da bi jih lahko učinkovito uporabili za raziskovanje. To naredimo tako, da za vsako obrobno regijo določimo, da število vsebovanih obrobnih točk pomnoženo z velikostjo celice, predstavlja njeno velikost. Če je ta manjša od premera robota, dano regijo izločimo iz nadaljnje obravnave, saj je premajhna, da bi jo robot obiskal, kot je prikazano na sliki 3.7. Preostale obrobne regije uporabimo kot podlago pri ustvarjanju izvirnih obrobij ter njihovih zamaknjenih kopij, kar je vidno s slike 3.8.



Slika 3.7: Filtrirane regije



Slika 3.8: Zamaknjene kopije

Objekt obrobje predstavljen v tabeli 3.2 z vrednostmi položaja, orientacijo in velikostjo predstavlja zgoščeno obliko obrobne regije. Te vrednosti pozneje uporabimo za izračun četrte vrednosti, cene raziskovanja, ki jo uporabimo za izbiro raziskovalnega cilja. Za velikost obrobja uporabimo že izračunano velikost obrobne regije. Položaj obrobja, ki mu pravimo tudi centroid, predstavlja potencialni raziskovalni cilj, ki ga moramo za potrebe vodenja robota hraniti v obliki točke oziroma vektorja. Izračunamo ga kot povprečen položaj obrobnih točk, ki sestavljajo obrobje. Orientacijo, podobno

kot položaj, dobimo s povprečjem oziroma normaliziranjem vsote orientacij obrobnih točk. Podobno kot orientacija točk, predstavlja orientacija obrobja smer proti najbližji neraziskani okolici. Vsako tako ustvarjeno izvirno obrobje dodamo na seznam obrobij, nato pa na njegovi podlagi ustvarimo dve zamaknjene kopije. Te ustvarjamo zato, ker se lahko zgodi, da se cilj, kot povprečni položaj obrobja, nahaja v neraziskanem prostoru v katerem robot ni zmožen načrtovati poti in ga tako zavrne kot nedosegljivega. Da bi se izognili izgubi potencialno dobrega raziskovalnega cilja, ustvarimo kopije danega obrobja, ki so zamknjena proti raziskanem delu prostora. Kopije nato dodam na seznam obrobij, ki se ga posreduje naslednji funkciji za izračun cene raziskovanja. Delovanje funkcije je bolj nazorno prikazano v pseudokodi algoritma 7.

spremenljivka	namen
`pozicija`	2D vektor, ki predstavlja x in y koordinate obrobja
`orientacija`	2D vektor, ki predstavlja orientacijo obrobja
`velikost`	predstavlja število obrobnih točk, ki sestavljajo obrobje
`cena`	predstavlja ceno raziskovanja danega orobja

Tabela 3.2: Objekt obrobje

Algorithm 7 *dobi_obrobja*

Input: seznam_regij

Output: seznam_obrobij

```
1: seznam_obrobij = nov seznam
2: for all regija r na seznamu_regij do
3:     if r.velikost >velikost_robota then
4:         vektor smer, položaj
5:         for all celica c v r do
6:             smer += c.smer
7:             položaj += c.položaj
8:         end for
9:         smer /= r.velikost
10:        položaj /= r.položaj
11:        for j=0 to 3 do
12:            nov_položaj = položaj + r.smer * j
13:            obrobje = novoobrobje( nov_položaj, r.smer, r.velikost )
14:            Enqueue( seznam_obrobij, obrobje )
15:        end for
16:    end if
17: end for
18: return seznam_obrobij
```

3.6 Funkcija OCENI OBROBJA

Naloga te funkcije je cenitev in sortiranje obrobij v skladu z vnaprej določeno strategijo raziskovanja. Kot lahko vidimo s pseudokode algoritma 8, funkcija v zanki za vsa obrobja na seznamu, tako izvirne kot zamaknjene kopije, izračuna ceno raziskovanja z rabo funkcije IZRAČUN CENE RAZISKOVANJA. Nato ta seznam ocenjenih obrobij po izračunanih cena raziskovanja tudi sortira, od najcenejšega, in s tem v skladu s strategijo raziskovanja najbolj primernega za raziskovanje, do najdražjega oziroma najslabšega za raziskova-

nje. Tako pripravljen seznam se kasneje v funkciji IZBERI CILJ uporabi za izbiro najbolj primernega raziskovalnega cilja. Da bi olajšali poznejšo izbiro raziskovalnih ciljev, tako predelan seznam obrobij nazadnje razporedimo po ceni raziskovanja od najnižje do najvišje.

Algorithm 8 *oceni_obrobje*

Input: seznam_obrobij

Output: seznam_obrobij

1: **for all** obrobje o na seznamu_obrobij **do**
2: *izračun_cene_raziskovanja*(o, položaj_robota, ločjivost_zemljevida)
3: **end for**
4: **return** seznam_ocenjenih_obrobij

3.6.1 Funkcija IZRAČUN CENE RAZISKOVANJA

Kot je razvidno že iz imena funkcije, je njen namen izračun cene raziskovanja za dano obrobje. Medtem ko je v Yamauchijevi izvirni različici [6] ceno raziskovanja enostavno določala le dolžina poti od robota do obrobja, v naši predelani različici za izračun cene raziskovanja poleg te uporabljamo še dve oceni, in sicer oceno razlike v orientaciji in oceno pridobitve informacij. Vse tri ocene osnovane na položaju, orientaciji in velikostjo obrobja, izračunanih med iskanjem obrobij v funkciji DOBI OBROBJA , izračunamo s funkcijami DOBI OCENO ODDALJENOSTI, DOBI OCENO SPREMEMBE ORIENTACIJE in DOBI OCENO PRIDOBITEV INFORMACIJ. Ker nismo več omejeni le na uporabo ene same ocene, na podlagi katere bi lahko sortirali seznam obrobij, lahko uporabimo poljubno kombinacijo vseh treh. V kolikšni meri upoštevamo posamezno oceno, določimo z utežmi. Ta princip izrabimo za določanje poljubne strategije raziskovanja našega robota. Izberemo lahko na primer, upoštevanje le ocene oddaljenosti in s tem simuliramo Yamauchijev pristop, enakomerno upoštevanje vseh ocen, ali pa katerokoli drugo kombinacijo. Ko so ocene utežene, jih je treba združiti v skupno ceno raziskovanja. Ker je naš namen karseda hitro raziskati čim večji del sveta in

robot porabi več časa za obračanje in potovanje, velja, da čim večji sta oceni oddaljenosti in spremembe orientacije, tem slabši sta za raziskovanje, zato ju prištejemo ceni raziskovanja. Na drugi strani pa uteženo oceno pridobitve informacij, za katero velja, da čim večja je, tem boljša je, saj predstavlja potencialno velikost novo raziskanega dela zemljevida, odštejemo od cene raziskovanja. Tako dobljeno ceno raziskovanja obrobja za lažjo rabo hranimo v samem obrobju. Bolj podrobno delovanje funkcije lahko vidimo v pseudokodi algoritma 9.

Algorithm 9 *izračun_cene_raziskovanja*

Input: obrobje, utež_oddaljenosti, utež_spremembe_smeri, utež_pridobitve_informacij

Output: cena

```
1: cena = 0
2: if utež_oddaljenosti ≠ 0 then
3:     cena += utež_oddaljenosti * dobi_oceno_oddaljenosti( obrobje.smer, položaj_robota )
4: end if
5: if utež_spremembe_smeri ≠ 0 then
6:     cena += utež_spremembe_smeri * dobi_oceno_spremembe_smeri( obrobje.položaj )
7: end if
8: if utež_pridobitve_informacij ≠ 0 then
9:     cena += - utež_pridobitve_informacij * dobi_oceno_pridobitve_informacij( obrobje.velikost, ločljivost )
10: end if
11: return cena
```

3.6.2 Funkcija DOBI OCENO ODDALJENOSTI

S to funkcijo, prikazano v algoritmu 10, izračunamo oceno potovanja izbranega obrobja. Ta predstavlja dolžino poti od trenutnega položaja robota do

izbranega obrobja, ki jo na cenovnem zemljevidu najde globalni načrtovalec poti. Daljša kot je pot, slabša je ocena obrobja, saj je treba dalj časa potovati do njega.

Algorithm 10 *dobi_oceno_oddaljenosti*

Input: položaj_obrobja

Output: ocena_oddaljenosti

1: **return** načrtovalec_poti.*do_cilja*(položaj_obrobja)

3.6.3 Funkcija DOBI OCENO SPREMEMBE ORIENTACIJE

Z *dobi oceno spremembe orientacije* predstavljamo, za koliko se mora robot obrniti, da bo pravilno orientiran, ko doseže obrobje. Funkcija, katere delovanje lahko vidimo na algoritmu 11, to oceno izračuna kot razliko med orientacijo robota in izbranega obrobja. Ker sta dani orientaciji v različnih oblikah – orientacija obrobja je v obliki vektorja, medtem ko je orientacija robota v stopinjah – je treba obe vrednosti spremeniti v skupno obliko in usmerjenost. To naredimo z arkustangens funkcijo *atan2*. Dobljeni vrednosti nato medsebojno odštejemo, izračunana razlika pa predstavlja oceno spremembe orientacije. Ta je slabša, tem večja je, saj to pomeni, da bo robot moral porabiti več časa za obračanje.

Algorithm 11 *dobi_oceno_spremembe_smeri*

Input: položaj_robota, smer_obrobja

Output: ocena_spremembe_smeri

1: smer_robota = položaj_robota.smer
2: atan2_robota = *atan*2(položaj_robota.y + *sin*(smer_robota), položaj_robota.x + *cos*(smer_robota))
3: atan2_obrobja = *atan*2(položaj_obrobja.x, položaj_obrobja.y)
4: ocena_spremembe_smeri = atan2_robota - atan2_obrobja
5: **return** ocena_spremembe_smeri

3.6.4 Funkcija DOBI OCENO PRIDOBLJENIH INFORMACIJ

S funkcijo *dobi oceno pridobljenih informacij* za dano obrobje izračunamo, koliko novih informacij bi pridobili, z raziskovanjem tega cilja. Ta ocena je enostavna, saj gre za velikost danega obrobja, prilagojeno velikosti raziskanega zemljevida, kar je razvidno z algoritma 12. Posledično obrobje, ki meri sto celic, na zemljevidu z ločljivostjo štiristo enot nima enake ocene kot na zemljevidu z ločljivostjo šeststo enot. Večja kot je ocena obrobja, tem boljša je, saj to pomeni, da z njenim raziskovanjem pridobimo več informacij in si tako izboljšamo zemljevid.

Algorithm 12 *dobi_oceno_pridobitve_informacij*

Input: velikost_obrobja, ločljivost_zemljevida

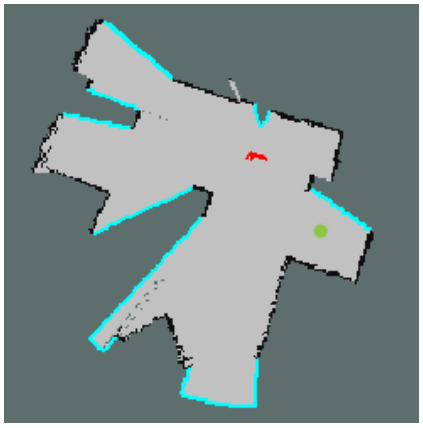
Output: ocena_pridobitve_informacij

1: **return** velikost_obrobja * ločljivost_zemljevida

3.7 Funkcija IZBERI CILJ

Ko od funkcije *oceni obrobja* dobimo sortiran seznam obrobij ocenjenih v skladu z izbrano strategijo, je potrebno izbrati cilj, ki je dosegljiv, oziroma ga je smiselno raziskati. Kot lahko vidimo s pseudokode algoritma 13, preverjanje začnemo tako, da s seznama vzamemo »najcenejši« možni cilj in se prvo prepričamo, da se ne nahaja v bližini elementov seznama prepovedanih ciljev. Med te spadajo že obiskani cilji ter cilji katerih raziskovanje je bilo prekinjeno, kot posledica nedosegljivosti, zaradi česa bi raziskovanje izbranega območja bilo nesmiselno. Nato z ROS-ovim globalnim načrtovalcem preverimo ali do izbranega cilja obstaja pot. V kolikor trenutni cilj ni blizu prepovedanih in do njega obstaja pot, ga izberemo za raziskovalni cilj. Primer tega vidimo na sliki 3.9 Tega nato višja funkcija *izberi raziskovalni cilj* posredujemo paketu za navigacijo, ki bo robota poskušal pripeljati do njega. V kolikor pa trenutni cilj enega od pogojev ne izpolnjuje, se ga zavrže in se

postopek preverjanja ponovi za naslednji »najcenejši« cilj s sortiranega seznama vzame naslednjega. Izbrane cilje se posreduje navigacijskim paketom ROS-a, ki poskusijo robota pripeljati do zastavljenega cilja, kot je prikazano na sliki 3.10. Ne glede na to ali robot uspešno doseže zastavljeni cilj ali pa je potovanje prekinjeno, ta cilj dodamo na seznam prepovedanih ciljev in ponovimo postopek izbiranja ciljev. Ta postopek raziskovanja v zanki ponavljamo, dokler nam ne zmanjka dosegljivih raziskovalnih ciljev in tako zaključimo z raziskovanjem.



Slika 3.9: Izbrani raziskovalni cilj

Slika 3.10: Premik do cilja

Algorithm 13 *izberi_dosegljiv_cilj*

Input: seznam_ciljev, prepovedan_seznam

Output: izbrani_raziskovalni_cilj

```
1: for all obrobje o na seznamu_ciljev do
2:     if o ni na prepovedanem_seznamu then
3:         if obstaja načrt od robota do obrobja then
4:             return o
5:         end if
6:     end if
7: end for
8: return null
```

Poglavje 4

Analiza rezultatov raziskovanj

Kot smo že omenili je namen tega diplomskega dela primerjava strategij izbire raziskovalnega cilja z namenom najti najboljšo, torej tisto, ki v čim krajšem času, čim bolj natančno razišče čim večji del prostora. V ta namen smo za vsak poskus merili raziskanost prostora, čas raziskovanja ter natančnost raziskanega zemljevida. Da smo lahko strategije primerno ocenili in primerjali, smo meritve opravili tako med raziskovanjem prostora, kjer smo lahko opazovali učinkovitost strategij, kot tudi ob zaključku raziskovanja, kjer smo lahko videli končne rezultate zaključenega raziskovanja. Raziskovanja so bila izvedena v prostorih Laboratorija za umetne vizualne spoznavne sisteme, kjer smo v ta namen postavili poligon s preprekami. Prostor s poligonom je viden na sliki 4.1. Pred začetkom vsakega poskusa raziskovanju prostora smo robota postavili na različni konec prostora ter ga obrnili v drugačno smer v primerjavi s prejšnjimi poskusi. S takimi naključnimi začetnimi postavitvami robota smo želeli preprečiti možnost, da bi katera od strategij lahko nepravično dosegla boljše rezultate, ker se je raziskovanje konstantno začelo v okolici, v kateri je izbrana strategija bolj uspešna od ostalih. Medtem ko je bilo potrebno za vsako strategijo opraviti 10 uspešnih raziskovanj, je med raziskovanjem občasno prihajalo do napak, zaradi katerih smo morali poskuse nekaterih strategij ponoviti tudi do 17 krat preden smo imeli dovolj podatkov za analizo.

Slika 4.1: Testni poligon

4.1 Strategije izbire raziskovalnih ciljev

Kot smo povedali že v prejšnjem poglavju strategijo raziskovanja določa strategija izbire raziskovalnih ciljev, ki je v celoti odvisna od vsote uteženih ocen raziskovanja; ocene oddaljenosti od robota do cilja, ocene spremembe orientacije med robotom in ciljem ter ocena pridobitve informacij. V kolikšni meri je katera ocena upoštevana pri izbiri raziskovalnega cilja in je tako določena raziskovalna strategija, pa določajo vrednosti uteži. Medtem ko algoritem omogoča uporabo vrednosti uteži med 0.0 in 1.0, je število možnih strategij, ki bi jih bilo potrebno testirati preveliko. Zato smo za potrebe diplomskega dela poenostavili nabor vrednosti uteži na 0 in 1 ter tako določili, da se posamezna ocena raziskovanja upošteva v celoti ali pa sploh ne. Tako zmanjšan nabor vednosti uteži dovoljuje le 8 kombinacij upoštevanja raziskovalnih ocen in s tem 8 raziskovalnih strategij, ki jih lahko vidimo v tabeli 4.1. Tri strategije, ki upoštevajo le eno izmed treh ocen raziskovanja, smatramo kot "osnovne". Z njimi želimo preveriti vpliv posamezne ocene na izbiro

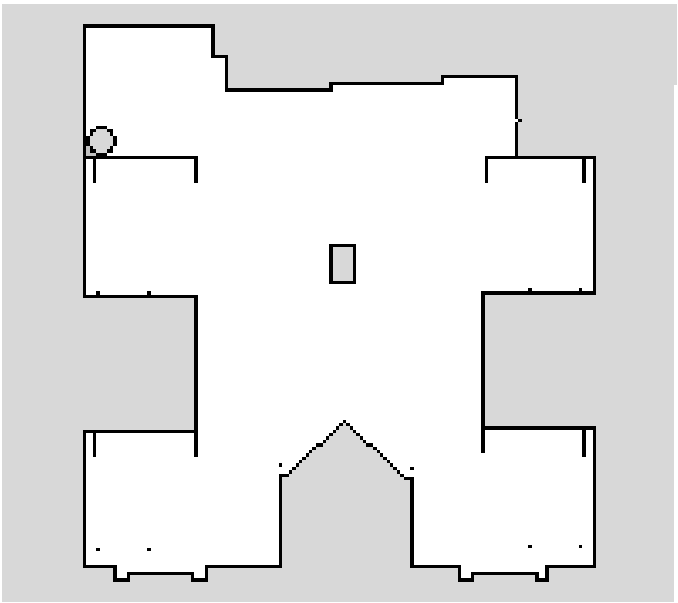
ciljev in s tem na proces raziskovanja. Naslednje štiri strategije pri izboru raziskovalnih ciljev upoštevajo dve ali pa vse tri ocene hkrati. Z njimi želimo preveriti, kako hkratno upoštevanje ocen vpliva na rezultate raziskovanja, torej ali jih izboljša ali poslabša. Zadnja strategija pri izračunu cen raziskovanja ciljev ne upošteva nobene izmed ocen, zaradi česa imajo vsi cilji enako ceno raziskovanja, 0. V tem primeru namesto uporabe cene raziskovanja za izbiro raziskovalnega cilja, le tega s seznama izberemo naključno. Medtem ko v tem primeru sicer ne moremo ugotoviti vpliv ocen, pa lahko ta pristop uporabimo kot kontrolni primer pri primerjavi učinkovitosti delovanja ostalih strategij.

strategija	ocena oddaljenosti	ocena spremembe orientacije	ocena pridobitve informacij
oddaljenost	X		
orientacija		X	
pridobitev informacij			X
oddaljenost & orientacija	X	X	
oddaljenost & pridobitev informacij	X		X
orientacija & pridobitev informacij		X	X
oddaljenost & orientacija & pridobitev informacij	X	X	X
naključna			

Tabela 4.1: Upoštevanje ocen raziskovanja za posamezno strategijo

4.2 Odstopanje zemljevida prostora

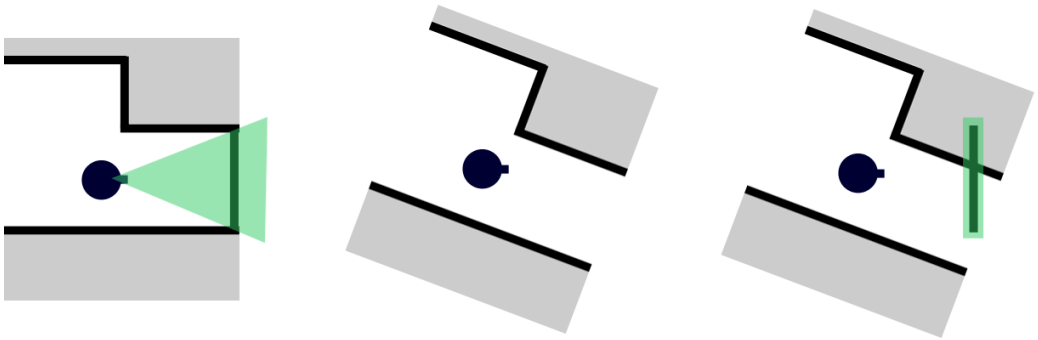
Zemljevid prostora je v tem diplomskem delu zelo pomemben podatek. Poleg enostavne naloge prikaza raziskanega dela prostora, smo ga uporabili predvsem za izračun raziskanosti prostora in natančnost te raziskanosti. Medtem ko lahko dobljene zemljevide preprosto primerjamo med seboj, smo za natančne meritve ter primerjavo z dejanskim prostorom potrebovali popolni zemljevid danega prostora. Slednjega smo sestavili tako, da smo najprej ročno vodili Turtlebota po prostoru in s paketom *gmapping* ustvarili karseda natančen zemljevid. Ker pa je ta imel manjše napake, smo ga pravilno orientirali in ga na podlagi izmerjenih dimenzij prostora v programu za obdelavo slik primerno popravili. Tako pripravljen idealni zemljevid prikazan na sliki 4.2 smo nato uporabili za natančno primerjavo z zemljevidi raziskovanja.



Slika 4.2: Idealni zemljevid

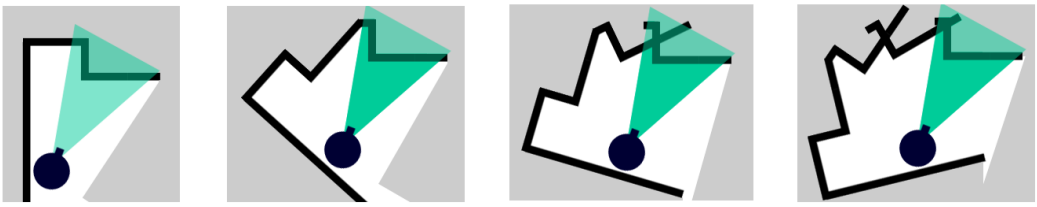
Medtem ko idealni zemljevid prikazuje dejanske dimenzije in razporejenost prostora, pa je zemljevid, ki ga dobimo na podlagi raziskovanja, v primerjavi z njim pogosto popačen. Zaradi napak pri odometriji (npr. zdrs koles) prihaja do napačne lokalizacije znotraj prostora, kar povzroča napake

pri kartiranju prostora. To se na zgrajenem zemljevidu kaže kot kartiranje prostega in oviranega prostora na napačne dele zemljevida, kvarjenje oblik ovir, kot je krivljenje ravnih sten, in podvajanje delov zemljevida, kot je prikazano na sliki 4.3. Če so te napake majhne, se jih večinoma vsaj delno odpravi z normalnim delovanjem paketa *gmapping* med raziskovanjem prostora. Čeprav te manjše napake kvarijo natančnost zemljevida, pa običajno ne predstavljajo večjih problemov pri navigaciji Turtlebota skozi prostor, zato jih dopuščamo.

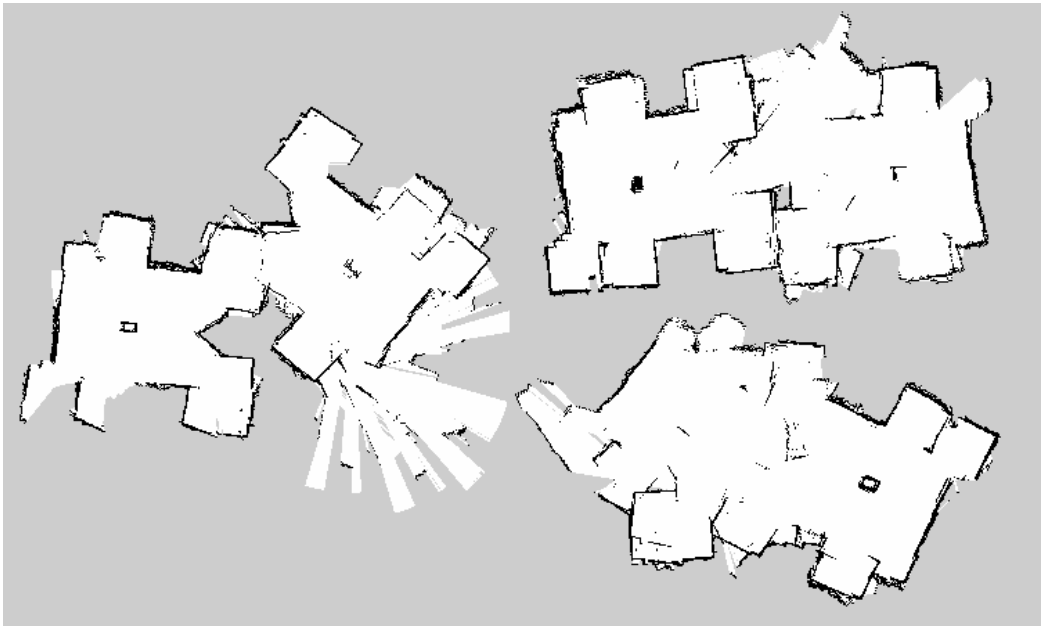


Slika 4.3: Nastanek manjše napake pri gradnji zemljevida

Če pa SLAM napak ne odpravi v zadostni meri, oziroma se te sčasoma nakopičijo, lahko to povzroča težave pri navigaciji robota in natančnosti zemljevida, in sicer do te mere, da zemljevid za naše potrebe ni več uporaben. Zaradi kopičenja napak pri lokalizaciji prihaja do izjemno napačnih kartiranj prostora. To povzroča napake pri navigaciji, saj robot načrtuje pot skozi prostor, za katerega misli, da je prost, v resnici pa je na tistem mestu ovira, v katero robot običajno trči. Pogosto se robot zaradi trka zatakne ob oviro, kar povzroča dodatne napake pri kartiranju, saj robot na podlagi odometrije sklepa, da se premika, medtem ko v resnici stoji na mestu. SLAM algoritem nato kartira isto zaznavo prostora na napačne dele zemljevida, ker zaradi napačne odometrije misli, da zavzema drug položaj, na način prikazan na sliki 4.4. Tako pokvarjenih in nenatančnih zemljevidov, prikazanih na sliki 4.5 ne moremo uporabiti za analizo, zato jih zavržemo.



Slika 4.4: Kopičenje napaka pri gradnji zemljevida



Slika 4.5: Primeri zavrženih popačenih zemljevidov

4.3 Analiza raziskanosti prostora

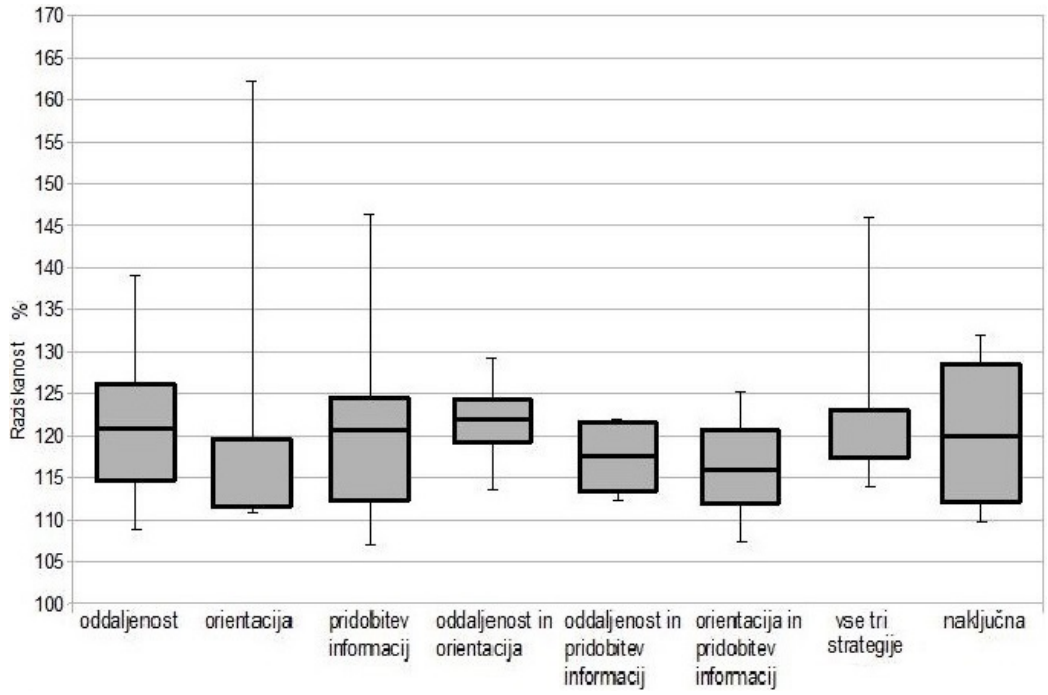
Kriterij raziskanosti nam pove, kolikšen del prostora je raziskan. Načeloma je ta kriterij nenatančen, saj naj bi robot vedno raziskal celoten prostor, vendar pa zaradi delovanja Turtlebota to včasih ni mogoče. Pri raziskovanju lahko določi, da kakšen cilj ni dosegljiv in zato ne razišče tistega dela prostora. Ker smo meritev poleg končne meritve raziskanosti podatke želeli uporabiti tudi dinamično za analizo hitrosti postopka raziskovanja, smo za mero raziskanosti uporabili število raziskanih, tako prostih kot oviranih, ce-

lic zemljevida. Na ta način smo se želeli izogniti uporabi procesno zahtevnih metod opravljanja meritev, ki bi s svojim delovanjem ovirale proces raziskovanja. Ko smo zastavili način merjenja raziskanosti smo predvidevali, da bo proces kartiranja deloval idealno, oziroma da bodo odgovorni paketi ROS-a pravilno kartirali vse zaznane podatke. Na ta način bi naš pristop merjenja pravilno beležil raziskanost, medtem ko bi bil ustvarjen zemljevid prostora pravilno poravnana z idealnim in tako imel 100% natančnost raziskanega. Med testiranjem pa smo ugotovili, da ustvarjeni zemljevidi zaradi prej omenjenih napak pri kartiranju nimajo predpostavljene 100% natančnosti, s tem pa tudi da izmerjena raziskanost ni predstavlja resnično stanje raziskanosti. Ker 100% natančnost zemljevidov ni bila več zagotovljena, se je pojavila potreba po opravljanju meritev in analizi natančnosti, poleg tega pa še problem, da ima lahko zemljevid, ne glede na to ali je prostor pravilno raziskan, enako število raziskanih celic kot idealni zemljevid. V praktično vseh primerih raziskovanja je zaradi napak v odometriji prihajalo tudi do napihovanja izdelanih zemljevidov, ter odebelitvijo robov zaznanih preprek. Tako imajo vsi pravilno raziskani zemljevidi, kljub pravilni obliki, približno 20% več raziskanih celic in s tem varljivo višjo mero raziskanosti v primerjavi z idealnim zemljevidom, katerega 18243 raziskanih celic smatramo kot 100% raziskanost zemljevida. Zaradi te nenatančnosti omenjeni kriterij uporabljamo bolj kot površno, informativno oceno pri primerjavi različnih strategij, kot pa absolutno mero raziskanosti in smatramo, da so rezultati boljši tem bližji so 100%.

4.3.1 Eksperimentalni rezultati

Kot lahko vidimo s slike 4.6, ki prikazuje dobljene eksperimentalne rezultate v bolj zgoščeni obliki škatlastega grafa, ter tabeli 4.2 imajo osnovne strategije večinoma večji razpon vrednosti ter slabše maksimalne vrednosti s približno enakimi povprečji kot naključna strategija, pa med njimi z najmanjšim odstopanjem od stoodstotne raziskanosti izstopa strategija orientacije (119,59%). V primerjavi z osnovnimi strategijami imajo sestavljene strategije, še posebej

tiste, ki upoštevajo le pare ocen hkrati, izjemno nizke standardne deviacije (približno 5%). Pri slednjih z najnižjima povprečnima vrednostma izmed vseh strategij izstopata kombinirani strategiji oddaljenosti in pridobitve informacij (117,50%) ter orientacije in pridobitve informacij (115,90%). Na podlagi tega sklepamo, da upoštevanje ocene pridobitve informacij, še posebej v kombinaciji z oceno orientacije, najbolje vpliva na raziskanost prostora.



Slika 4.6: Graf postopka raziskovanja naključne strategije

strategija	povprečje [%]	standardna diviacija [%]
oddaljenost	119,59	15,17
orientacija	120,60	11,66
pridobitev informacij	120,60	11,66
oddaljenost & orientacija	121,91	5,02
oddaljenost & pridobitev informacij	117,50	4,22
orientacija & pridobitev informacij	115,90	7,73
oddaljenost & orientacija & pridobitev informacij	123,07	9,44
naključna	119,89	9,06

Tabela 4.2: Raziskanost prostora

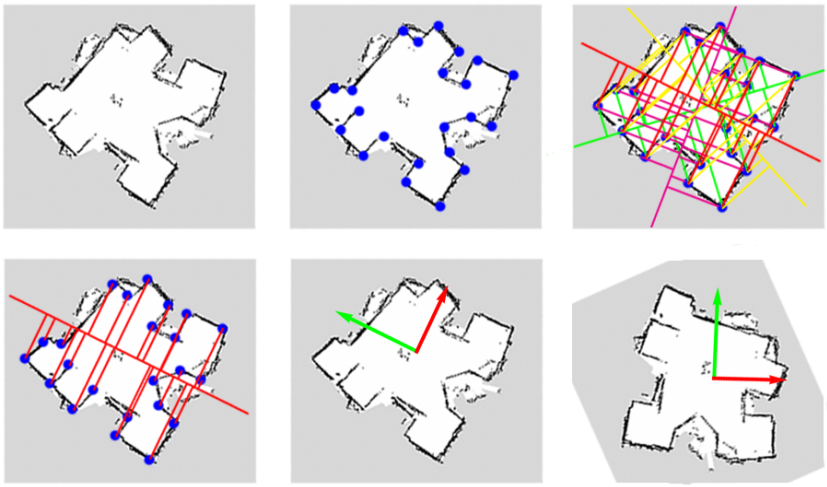
4.4 Analiza natančnosti

Natančnost je ena bolj pomembnih lastnosti zemljevida, saj robotu omogoča pravilno lokalizacijo v prostoru in navigiranje po njem. Kot smo omenili že pri analizi raziskanosti, smo meritve in primerjave natančnosti ustvarjenih zemljevidov začeli izvajati, ko se je predpostavka, da bo proces sestavljanja zagotovil 100% natančnost kartiranja, izkazal za zmotno. Za namen analize podatkov robot vsako raziskovanje začne na drugem položaju in z drugačno orientacijo. Ker se začetni položaj in orientacija Turtlebota uporabita kot izhodiščna točka koordinatnega sistema zemljevida, ima vsak zemljevid drugačno orientacijo ter zamik, zato ju je za namen primerjave natančnosti treba karseda natančno poravnati z zemljevidom. To naredimo

z dvema algoritmoma.

4.4.1 PCA

Z metodo analize osnovnih komponent (angl. principal component analysis) ali skrajšano PCA [19] najprej ugotovimo približno orientacijo idealnega zemljevida in izdelanih zemljevidov. Delovanje PCA je prikazano na sliki 4.7. Ta na danem zemljevidu poišče značilne točke, ponavadi robove ali kote, na sliki označene kot modre točke, ki jih nato uporabi za analizo razpršenost danih podatkov, z namenom poiskati smer v kateri je razpršenost točk največja. To poišče z eigenvektorjem, katerega smer predstavlja iskano smer, velikost (eigenvrednost) pa mero največje razpršenosti podatkov zemljevida. Skupaj z drugim eigenvekotorjem, ki je pravokoten na prvega, tvorita nov koordinatni sistem, prek katerega lahko ustvarimo transformacijsko matriko, s katero zemljevid preslikamo v xy koordinatni sistem. Na ta način lahko vse zemljevide poravnamo na približno isti položaj in podobno orientacijo, saj zaradi njihove medsebojne podobnosti sklepamo, da imajo podobno razpršenost značilnih točk in s tem določene eigenvektorje.



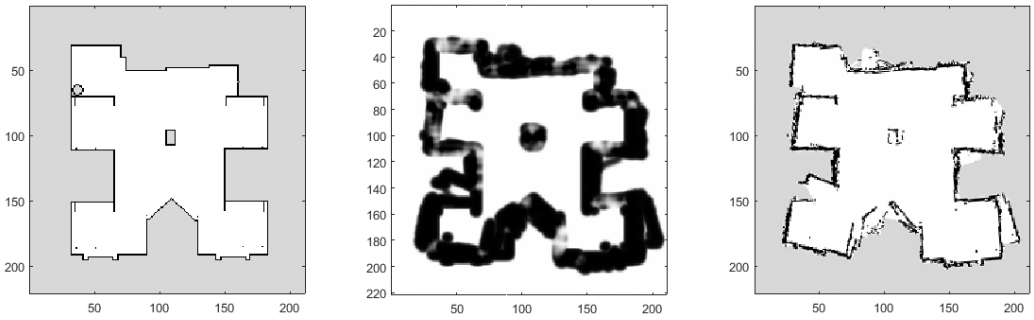
Slika 4.7: Delovanje PCA

4.4.2 ICP

Tako grobo poravnane zemljevide bolj natančno poravnamo z uporabo ICP algoritma [17]. Ta se uporablja za zmanjševanje razlik med dvema skupinama točk. Za potrebe delovanja algoritma ta na obeh zemljevidih najprej določi skupine značilnih točk, ponavadi robove ali kote, nato pa eno skupino uporabi kot referenco, medtem ko drugo transformira, da čim bolj sovpada z referenco. Na ta način algoritem iterativno popravlja transformacijo, ki je potrebna za zmanjševanje razdalje med skupinama točk, dokler ne doseže najmanjše možne.

4.4.3 SSIM

Nazadnje poravnane izdelane zemljevide primerjamo z idealnim zemljevidom z uporabo SSIM (ang. structural similarity) ali indeksa strukturne podobnosti [16]. Ta temelji na ideji, da je človeški vizualni sistem sposoben procesirati strukturne informacije slike, na podlagi katerih poskuša meriti razlike med slikama. Uporabljamo ga za izračun mere podobnosti ene slike v primerjavi z drugo, za katero predpostavljamo, da je popolna. Deluje tako, da na višjem nivoju meri spremembe v svetilnosti, kontrastu in strukturi slike, pri čemer je svetilnost modelirana kot povprečna intenziteta slikovnega elementa, kontrast kot razlika med referenčno in popačeno sliko ter struktura kot križna korelacija med obema slikama. Dobljene vrednosti so nato povprečene in združene v indeks z razpon vrednosti od 0.0 do 1.0, oziroma od 0% do 100%, ki nam pove mero podobnosti med slikama, kjer 100% pomeni da sta sliki identični, 0% pa da sta popolnoma različni. Primer delovanja si lahko ogledamo na sliki 4.8, kjer je na levi strani popolna slika idealni zemljevid, na desni z avtonomni raziskovanjem ustvarjen zemljevid, v sredini pa SSIM razlika.



Slika 4.8: Primerjava zemljevidov s SSIM

4.4.4 Eksperimentalni rezultati

Z uporabo prej omenjenega postopka primerjave ustvarjenih zemljevidov s idealnim smo dobili rezultate predstavljene v tabeli 4.3. Kot lahko vidimo se celoten razpon natančnosti raziskovalnih strategij giblje med 63% in 77%, s povprečji med 70–73%. S takih rezultatov lahko razberemo, da medtem ko imajo avtonomno zgrajeni zemljevidi sicer približno enake dimenzije, razporeditev in obliko kot idealni zemljevid, s katerim jih primerjamo, imajo vseeno preveč manjših napak, ki preprečujejo boljšo podobnst. Medtem ko med strategijami s tega vidika ni opaziti večjih razlik, na podlagi katerih bi lahko z gotovostjo trdili, da je določena strategija pokazala izredno boljše rezultate, pa vseeno obstajajo manjše razlike v učinkovitosti in konsistentnosti med posameznimi strategijami. Zaradi uporabe sosednjih delov raziskanega prostora kot reference SLAM algoritma za pravilno dodajanje novih podatkov, smo pričakovali, da bodo strategije, ki robota vodijo skozi že raziskane dele prostora, ustvarile bolj natančne zemljevide. V skladu s pričakovanji imajo v primerjavi z naključno strategijo boljša povprečja le strategije, ki upoštevajo oceno spremembe orientacije in pridobitve informacij, torej obe osnovni strategiji ter njuna kombinirana strategija. Vse ostale strategije pa imajo v primerjavi z naključno strategijo slabše povprečne vrednosti in upoštevajo oceno oddaljenosti. Na podlagi teh rezultatov sklepamo, da oceni orientacije in pridobitve informacij, podobno kot naključna strategija, z bolj "kaotičnim" vodenjem robota, pozitivno vplivata na natančnost izdelanih

zemljevidov, medtem ko jo bolj "usmerjena" ocena oddaljenosti kvari.

strategija	povprečje [%]	standardna diviacija [%]
oddaljenost	70,85	3,71
orientacija	72,54	2,43
pridobitev informacij	72,98	4,17
oddaljenost & orientacija	71,37	2,66
oddaljenost & pridobitev informacij	69,96	2,51
orientacija & pridobitev informacij	72,11	2,69
oddaljenost & orientacija & pridobitev informacij	70,70	1,70
naključna	71,85	2,91

Tabela 4.3: Natančnost ustvarjenih zemljevidov

4.5 Analiza časa raziskovanja

Kriterij časa raziskovanja uporabljamo za primerjanje hitrosti postopkov raziskovanja med strategijami, pri čemer je ta boljši čim krajši, oziroma hitrejši je. Čas raziskovanja merimo od začetka procesa, ko se robot prvič zavrti in zazna okolico, do trenutka ko ne najde več nobenega dosegljivega raziskovalnega cilja in zaključi z raziskovanjem prostora. V idealnih razmerah bi k času raziskovanja šteli le zaznavanje okolice in neprekinjeno potovanje do cilja. Vendar pa v realni situaciji oceno časa raziskovanja kvari delovanje paketov ROS-a, ki so potrebni za raziskovanje. Ti dodajo zamude k meritvi časa, zaradi čakanja robota na boljši načrt premikanja, oziroma zaradi

odstranjevanja nedosegljivih ciljev in preverjanja dosegljivosti cilja. Medtem ko dobljeni rezultati zaradi teh zamud ne prikazujejo idealnih časov raziskovanja, kot bi jih dobili, če bi teste opravili v simulacijah, kjer teh zamud ni, in s tem kvarijo natančnost ter zanesljivost ocene časa raziskovanja prostora, pa vseeno prikazujejo realne podatke raziskovanja v pravega okolja s pravim robotom. Čeprav bi pri izdelavi ponavadi raje imeli natančen in kar se da popoln, kot pa hitro zgrajen zemljevid in bi zaradi tega sklepali, da je kriterij časa raziskovanja sekundaren v primerjavi z natančnostjo in raziskanostjo, pa so ti v našem primeru enakovredni. Zaradi popravljanja napak kartiranja ali napačnega raziskovanja, ki sledi zaradi nepopravljene napake, se namreč čas raziskovanja posledično podaljša, kar lahko povzroči nove napake in ponovno podaljša čas raziskovanja itd. Tak cikel ponavadi negativno vpliva na natančnost ter raziskanost zemljevida. Tako so kriterij časa raziskovanja kot tudi dobljeni rezultati ne zgolj zaradi splošne želje po hitrejšemu raziskovanju prostora, temveč tudi zaradi manjšanja verjetnost nastanka napak zemljevida.

4.5.1 Eksperimentalni rezultati

Na podlagi zgoraj opisanega postopka smo dobili rezultate prikazane v tabeli 4.4. Najkrajši čas raziskovanja smo zaradi ideje, da bolj usmerjeno raziskovanje prispeva h krajšemu času, pričakovali od osnovnih strategij oddaljenosti in pridobitve informacij ter njune kombinirane strategije, slabše rezultate pa od strategij, ki pri izbiri raziskovalnih ciljev upoštevajo oceno orientacije, kar bi povzročilo daljše in bolj kaotično oziroma neusmerjeno raziskovanje. Najslabše rezultate smo pričakovali od kombinirane strategije orientacije in pridobitve informacij. V skladu s pričakovanji ima kombinirana strategija oddaljenosti in pridobitve informacij, enega najkrajših povprečnih raziskovalnih časov (18:36 min), ki pa je pri večini ostalih strategij, vključno z osnovnima strategijama oddaljenosti in pridobitve informacij, slabši od povprečja naključne strategije (17:18 min). Edini strategiji z boljšim povprečnim časom kot naključna strategija in z manjšim standardnim odklonom sta osnovna

strategija	povprečje [min]	standardna diviacija [min]
oddaljenost	20:06	6:12
orientacija	16:36	5:18
pridobitev informacij	20:18	6:36
oddaljenost & orientacija	23:36	7:00
oddaljenost & pridobitev informacij	18:36	7:36
orientacija & pridobitev informacij	16:12	4:24
oddaljenost & orientacija & pridobitev informacij	20:36	5:30
naključna	17:18	9:18

Tabela 4.4: Čas raziskovanja prostora

strategija orientacije (16:36 min) ter kombinirana strategija orientacije in pridobitve informacij (16:12 min). Zato sklepamo, da medtem ko upoštevanje ocene orientacije načeloma zmanjšuje raziskovalni čas, ga ocena oddaljenosti podaljšuje.

4.6 Primerjava postopkov raziskovanja

Kriterij raziskanosti prostora in čas raziskovanja v kombinaciji uporabimo za prikaz postopka raziskovanja oziroma mero raziskanosti skozi čas. Na ta način želimo preveriti, kako hitro posamezna strategija raziskuje prostor, oziroma kako se obnaša. Strategije smo primerjali na podlagi oblike grafov, ker pa je skupni prikaz vseh desetih procesov raziskovanja zaradi prepletenosti in gostote podatkov nerazumljiv, smo se odločili za prikaz vrednosti škatlastega

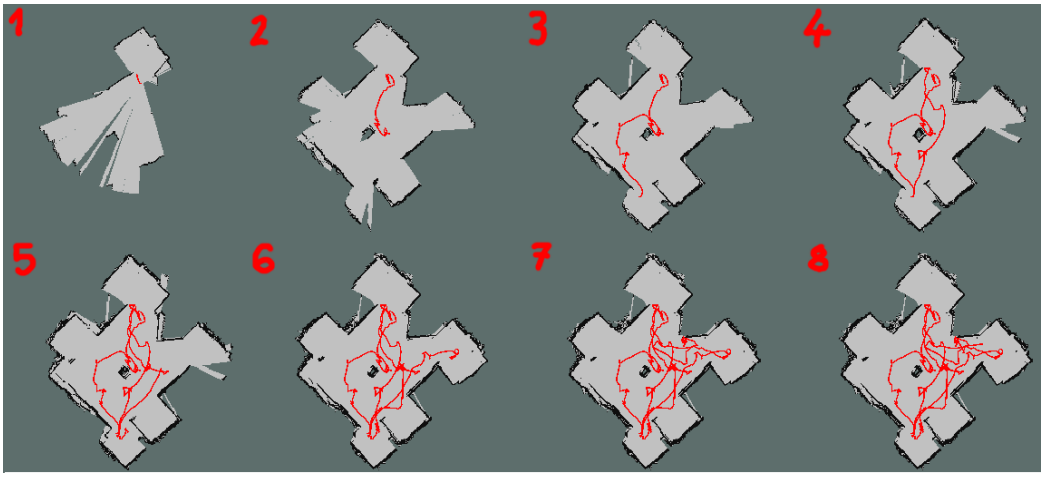
diagrama (angl. boxplot) te skupine procesov v obliki linearnega grafa. Te vrednosti poleg povprečja vsebujejo še najmanjšo in največjo vrednost ter prvi in tretji kvartal te skupine procesov. Za primerjavo raziskovalnih strategij smo raziskovali prostor z večjim osrednjim delom in manjšimi obrobni prostori. Ker robot pri procesu raziskovanja začne ali pa se iz obrobnega dela hitro premakne v osrednji del, iz katerega vidi večino prostora, raziskanost ne glede na izbrano strategijo hitro naraste. Temu nato sledi raziskovanje ostalih delov prostora, ki je v primerjavi z začetnim počasnejše. Zaradi te oblike prostora ima graf postopka raziskovanja obliko logaritemske funkcije. Kljub tej skupni obliki pa obstajajo razlike med grafi različnih strategij, kar je razvidno iz strmosti naraščanja in odstopanj od povprečja. Medtem ko so odstopanja od povprečne vrednosti strnjena na začetku, ko je prostor neraziskan, in koncu grafa, ko je v celoti raziskan, pa so zanimive vmesne vrednosti, ki nam povejo več o konsistentnosti posameznih strategij. Tako lahko v primeru, ko graf raste relativno hitro, sklepamo, da strategija najprej grobo razišče večino prostora in nato postopoma razišče preostale dele prostora ter na zemljevid dodaja podrobnosti. V nasprotnem primeru, ko graf raste relativno počasi, pa lahko sklepamo, da je to zaradi počasnejšega postopnega in podrobnejšega raziskovanja posameznih delov prostora. Medtem ko se zaradi preprostega načina merjenja raziskanosti prostora najvišje vrednosti posameznih strategij gibljejo od 120% pa vse do 150%, lahko s tabele 4.5, ki prikazuje hitrost raziskovanja, razberemo, da jih večina doseže 100% raziskanost, oziroma 18243 raziskanih celic, kolikor jih ima idealni zemljevid prostora, v povprečno 5,5–6 minutah. Tako imajo v primerjavi z ostalimi naključna strategija s povprečno 4 minutami, osnovni strategiji, ki upoštevata oceno orientacije ali pridobitve informacij, s 3–4 minutami, ter njuna kombinirana strategija s 3 minutami do 100% raziskanosti najboljše hitrosti raziskovanja.

strategija	povprečje [min/100%]
naključna	3:55
oddaljenost	6:23
orientacija	4:12
pridobitev informacij	3:22
oddaljenost & orientacija	5:14
oddaljenost & pridobitev informacij	6:09
orientacija & pridobitev informacij	3:18
oddaljenost & orientacija & pridobitev informacij	5:21

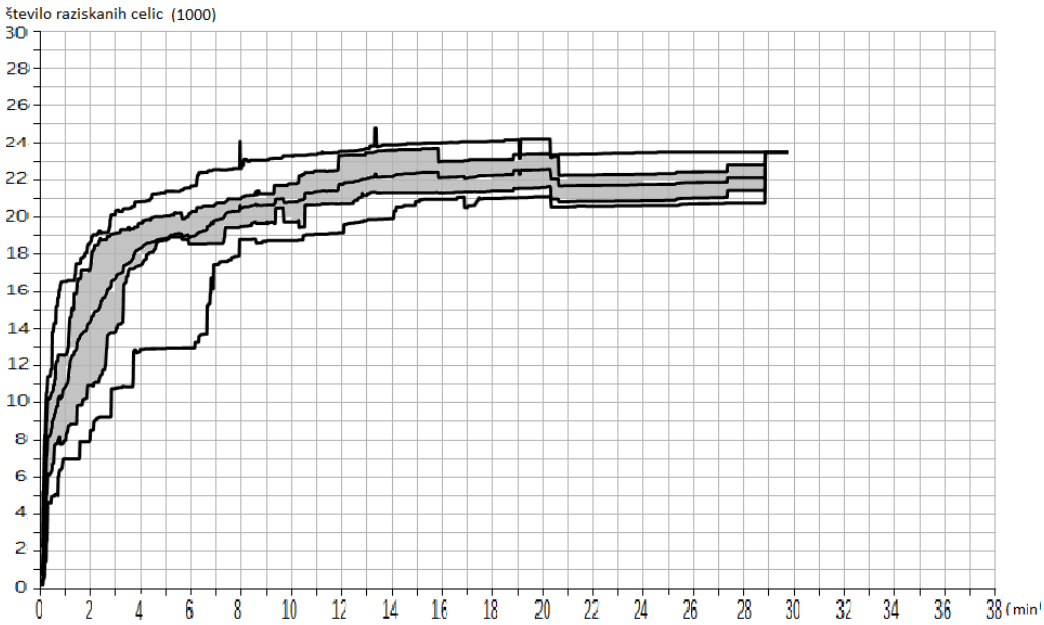
Tabela 4.5: Hitrost raziskovanja prostora

4.6.1 Naključna strategija

Strategijo naključne izbire raziskovalnega cilja uporabljamo predvsem za primerjavo učinkovitosti ostalih strategij po vseh kriterijih. Njena pot raziskovanja je zaradi načina izbire raziskovalnih ciljev dokaj kaotična, neusmerjena in nestalna, zato robot večkrat potuje v nasprotne dele prostora ter se nato vrne v že raziskan prostor, kar je razvidno s slike 4.9. Zaradi takšnega raziskovanja doseže 100% raziskanost oziroma 18243 raziskanih celic zemljevida, kolikor jih ima idealni zemljevid, v povprečno 3:55 minutah. Kot prikazuje slika 4.10 traja celotno raziskovanja prostora povprečno 17:18 minut z odstopanjem 9:18 minut, v katerem povprečno razišče med 20230 do 23500 celic zemljevida, oziroma doseže povprečno raziskanost 119,89% z odstopanjem 9,06%. Na podlagi rezultatov vseh kriterijev lahko rečemo, da je naključni izbor raziskovalnega cilja s seznama ciljev presenetljivo učinkovita strategija.



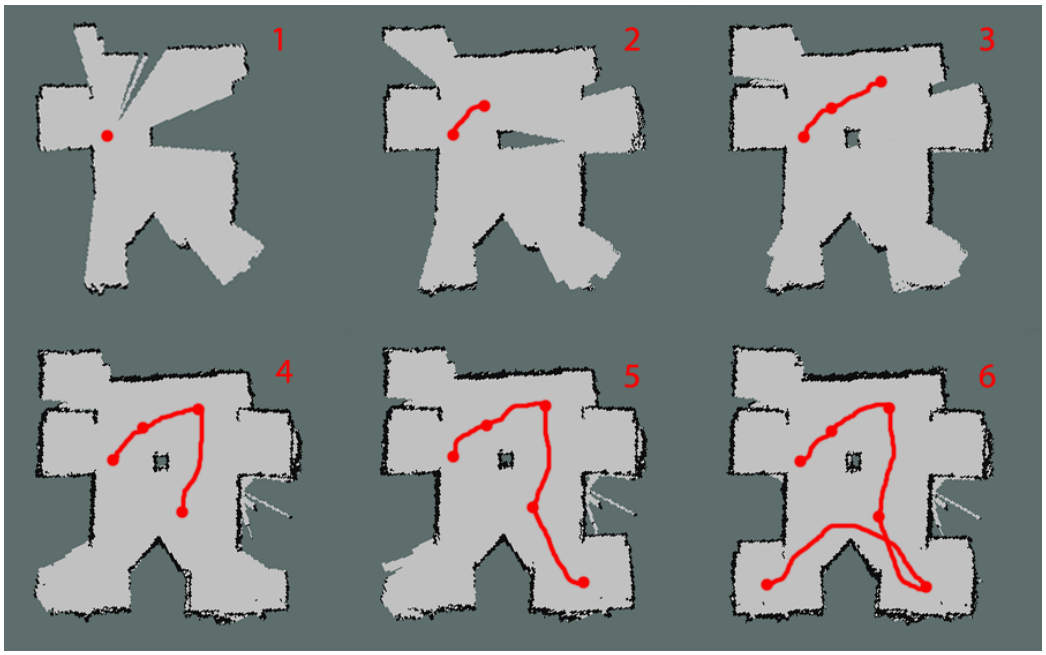
Slika 4.9: Potek raziskovanja nakjučne strategije



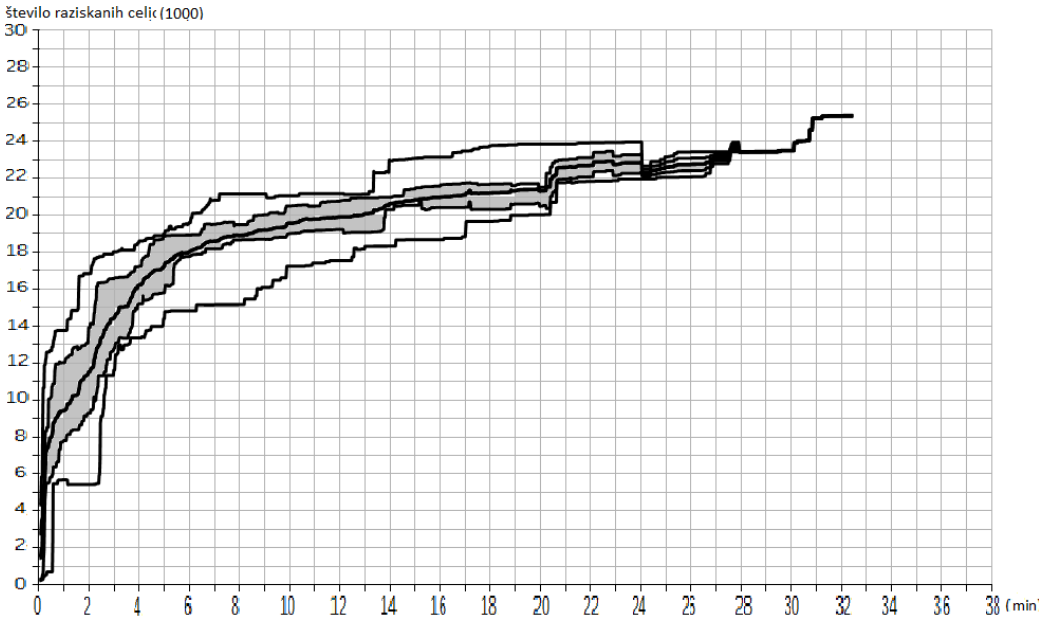
Slika 4.10: Graf postopka raziskovanja naključne strategije

4.6.2 Strategija oddaljenosti

Strategija izbire raziskovalnih ciljev na podlagi oddaljenosti od robota je bila prvotno predstavljena skupaj s konceptom uporabe obrobij kot raziskovalnih ciljev v [6]. Zaradi izbire najbližjih raziskovalnih ciljev smo pričakovali postopno raziskovanje prostora, kjer robot razišče vse znotraj določenega dela okolja, preden nadaljuje z raziskovanjem preostalega. Kot lahko vidimo na sliki 4.11, je pot raziskovanja zaradi tega bolj osredotočena in brez večjih odstopanj. Medtem ko smo za to strategijo zaradi osredotočenega raziskovanja pričakovali dobre rezultate, pa se je, kot lahko razberemo z grafa postopka raziskovanja 4.12, s počasnejšim postopkom raziskovanja 6:23 minut do 100% raziskanosti, podobo kot pri ostalih kriterijih slabo odrezala. Zaradi tega kot tudi ostalih slabih rezultatov je strategija izbora raziskovalnega cilja na podlagi ocene oddaljenosti najslabša osnovna strategija, njen slab vpliv, oziroma vpliv ocene oddaljenosti pa je razviden tudi pri kombiniranih strategijah, ki jo uporabljajo.



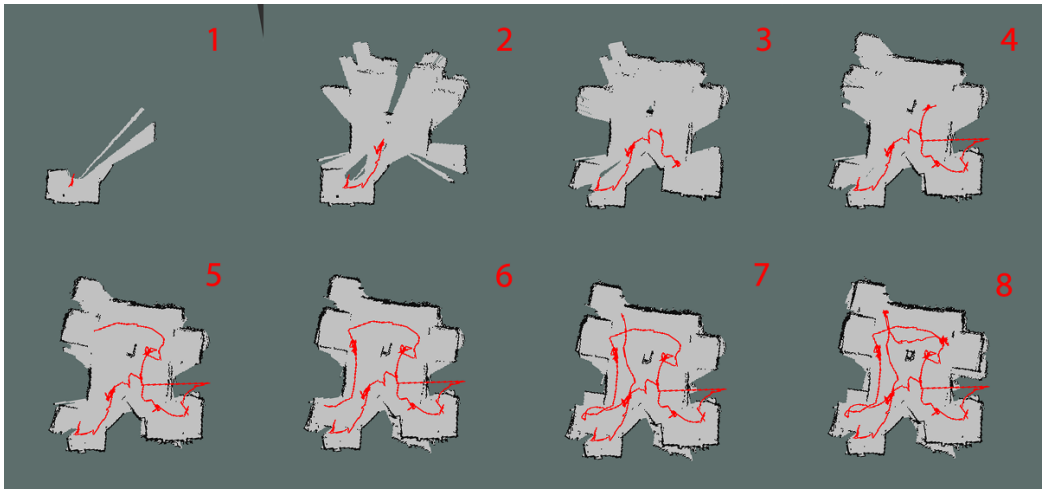
Slika 4.11: Potek raziskovanja strategije oddaljenost



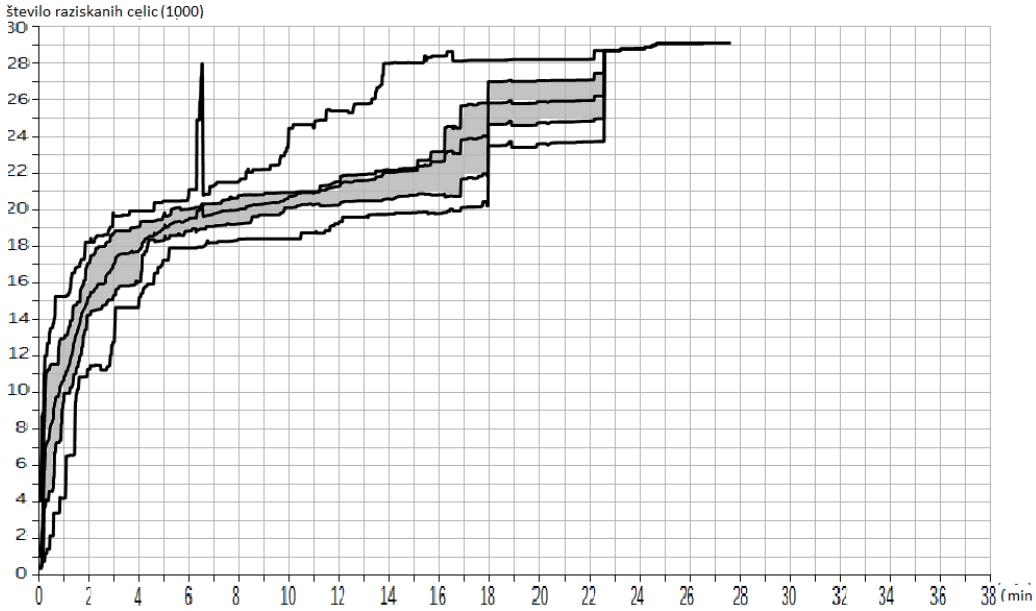
Slika 4.12: Graf postopka raziskovanja strategije oddaljenosti

4.6.3 Strategija orientacije

Za razliko od strategije oddaljenosti ta strategija ne upošteva, kje se nahaja raziskovalni cilj, temveč kako podobna je njegova orientacija v primerjavi z orientacijo robota. Ker pa imajo bližnja obrobja pogosto večjo razliko v orientaciji v primerjavi s ciljem na drugi strani prostora, je enako verjetno, da bo robot raziskoval v isti smeri ali pa se odpravil na drug konec prostora, kar je opazno slike 4.13, ki prikazuje pot raziskovanja. Kot lahko razberemo s slike 4.14, ki prikazuje graf postopka raziskovanja, je začetni proces raziskovanja s povprečnimi 4:12 minutami delovanja do raziskanosti 18243 celic zemljevida hiter. Medtem ko je s hitrostjo raziskovanja primerljiv z naključno strategijo, pa po ostalih kriterijih prekaša, ne le njo, temveč tudi večino ostalih strategij. Z najboljšimi povprečji in najmanjšimi odkloni pri raziskovalnem času, raziskanostjo in natančnostjo je to najboljša izmed osnovnih strategij ter primer dobrega vpliva ocene spremembe orientacije.



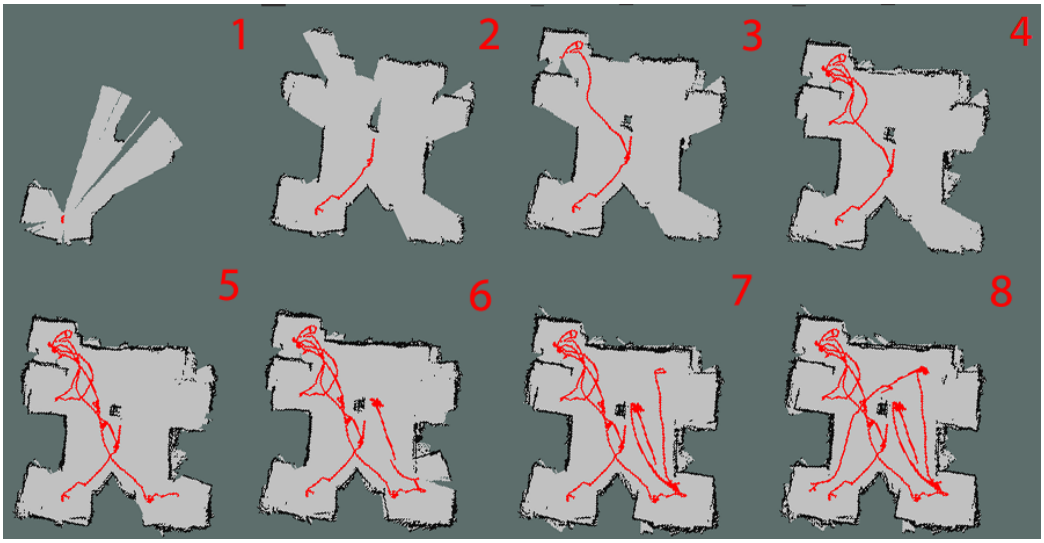
Slika 4.13: Potek raziskovanja strategije orientacije



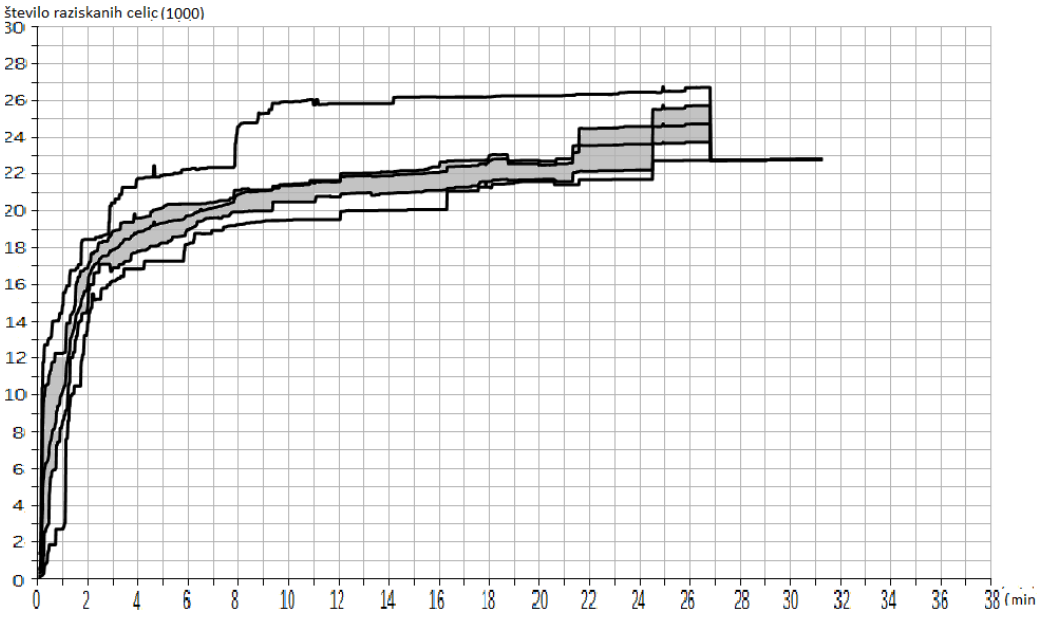
Slika 4.14: Graf postopka raziskovanja strategije orientacije

4.6.4 Strategija pridobitve informacij

Podobno kot naključna strategija in strategija orientacije tudi ta strategija razišče prostor dokaj kaotično, kar lahko vidimo slike 4.15, saj namesto bližine za izbor cilja upošteva oceno pridobitve informacij, ki temelji na velikosti obrobij. Zaradi takega delovanja strategije smo pričakovali krajši čas raziskovanja v primerjavi s strategijo orientacije ter daljšega kot pri strategiji oddaljenosti. Medtem ko se strategija, ki izbira raziskovalne cilje na podlagi ocene pridobitve informacij, s slabšimi povprečji in manjšimi odkloni v primerjavi z naključno strategijo pri kriteriju časa raziskovanja in raziskanostjo prostora ne loči dosti od strategije oddaljenosti, pa se lahko pohvali z drugo najvišjo stopnjo natančnosti. Kot lahko razberemo s grafa postopka raziskovanja 4.16 ima z raziskanimi 18243 celicami v 3:22 minutah najhitrejšim povprečnim začetni proces raziskovanja. Tako se strategija izkaže kot druga najboljša osnovna strategija, z njo pa tudi ocena pridobitve informacij.



Slika 4.15: Potek raziskovanja strategije pridobitve informacij

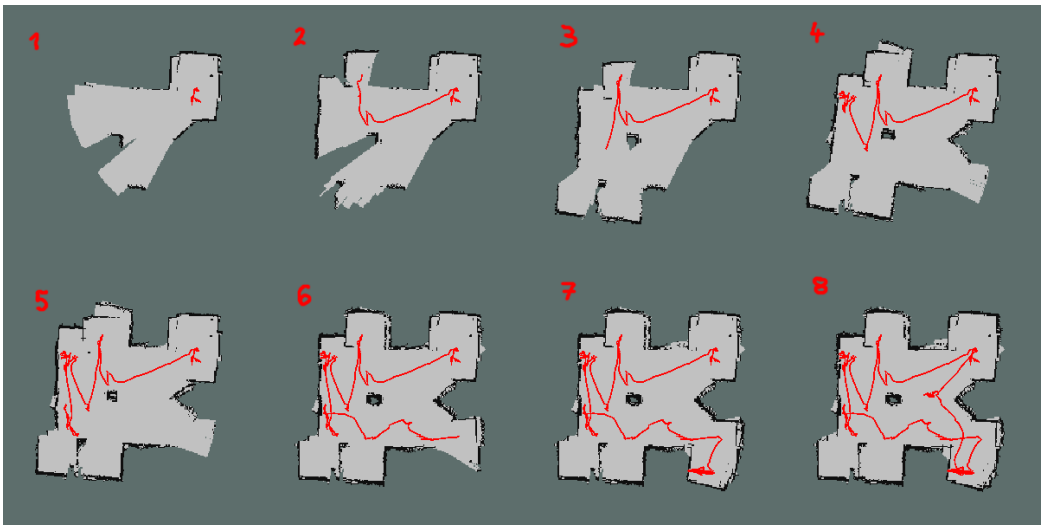


Slika 4.16: Graf postopka raziskovanja strategije pridobitve informacij

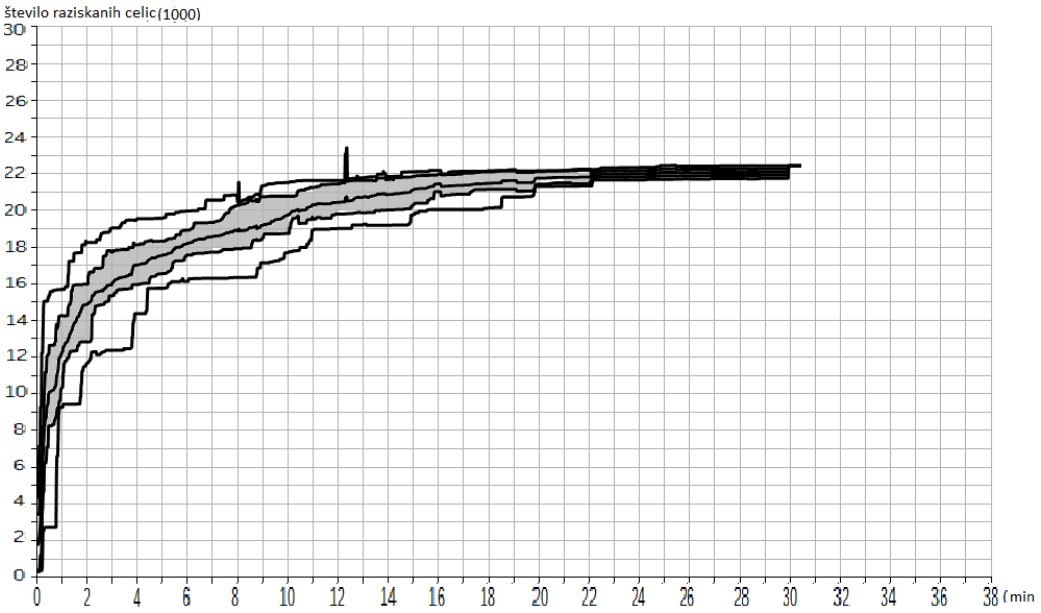
4.6.5 Strategija oddaljenosti in pridobitve informacij

Ta strategija pri izbiri raziskovalnih ciljev v enaki hkrati upošteva oceno oddaljenosti robota do cilja in oceno pridobitve informacij. Tako izmed najbližjih ciljev izbira največje, kar bi pomenilo neke vrste kombinacijo hitrega in natančnega raziskovanja, zaradi česa smo od ste strategije pričakovali najboljše rezultate. Pod vplivom ocen pridobitve informacij strategija ponovno razišče del prostora in občasno deluje kaotično, vendar pa jo vpliv strategije oddaljenosti kroti in skrbi za bolj usmerjeno raziskovanje. Kot lahko vidimo s slike 4.17, ki prikazuje potek raziskovanja, je pot raziskovanja krajša in bolj določena od poti same strategije pridobitve informacij, saj jo popravlja vpliv strategije oddaljenosti. Temu primerno postopek raziskovanja kot kompromis med osnovnima strategijama kaže povprečno učinkovitost obeh strategij. S slike, ki prikazuje graf postopka raziskovanja vidimo, da strategija raziskanost 18243, oziroma 100% količine celic idealnega zemljevida doseže v povprečno 6:09 minutah. Tako je bolj konsistentna in hitrejša od strategije oddaljenosti,

a hkrati počasnejša od strategije pridobitve informacij. Sodeč po povprečnih vrednostih in odstopanjih časov raziskovanja in natančnosti, so rezultati te kombinirane strategije z izjemo boljše raziskanosti zelo podobni rezultatom strategiji naključne izbire raziskovalnih ciljev.



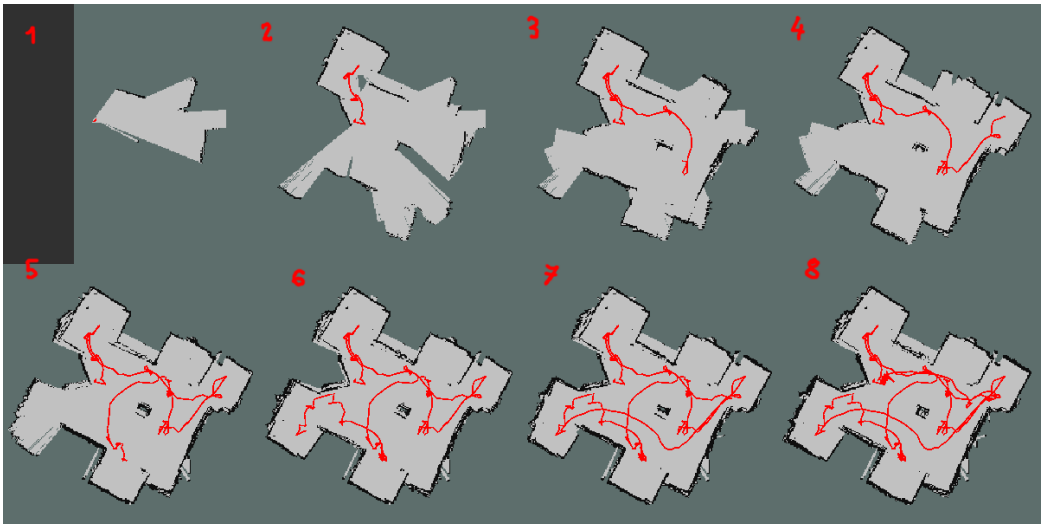
Slika 4.17: Potek raziskovanja strategije oddaljenosti in pridobitve informacij



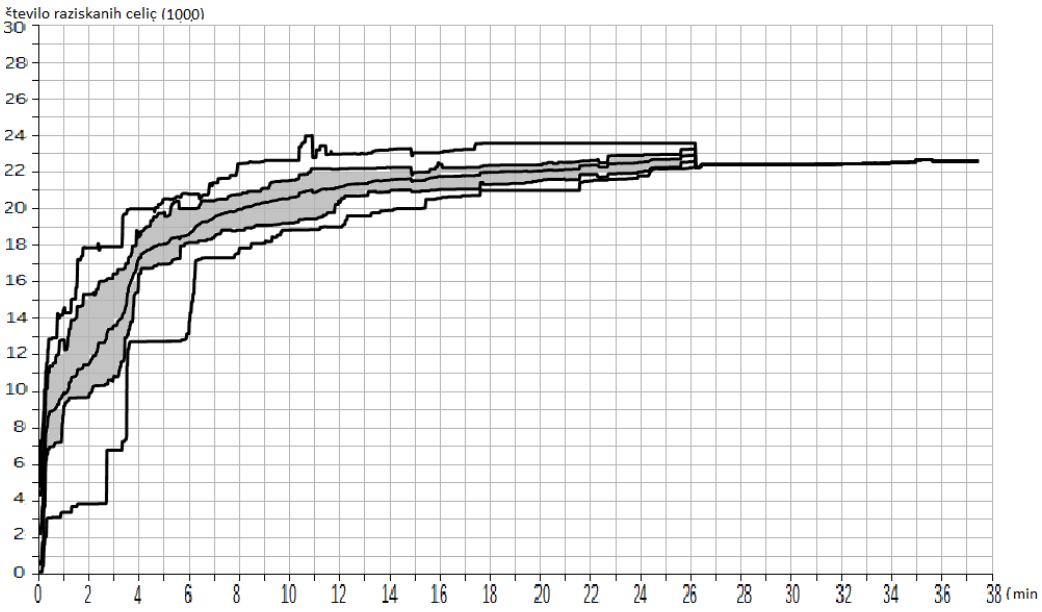
Slika 4.18: Graf postopka raziskovanja strategije oddaljenosti in pridobitve informacij

4.6.6 Strategija oddaljenosti in orientacije

Postopek raziskovanja s strategijo upoštevanja kombinacije ocene spremembe orientacije in ocene oddaljenosti do cilja zaradi izbiranja bližnjih, podobno orientiranih raziskovalnih ciljev poteka v približno isto smer. Vendar pa zaradi vpliva ocene orientacije pogosto zaide v že raziskan del prostora, kar v nasprotju s pričakovanji izjemno podaljša pot raziskovanja. Primer takega raziskovanja lahko vidimo na sliki 4.19. Kot lahko vidimo s slike **??**, ki prikazuje graf postopka raziskovanja, doseže povprečno raziskanost 18243 celic, kolikor jih je v 100% raziskanem idealnem zemljevidu v 5:14 minutah, zaradi česa je en slabših raziskovalnih procesov. Tako je ta kombinirana strategija kljub nizkim odstopanji od povprečnih vrednosti, z najslabšimi povprečnimi vrednosti kriterijev časa raziskovanja, natančnosti ustvarjenih zemljevidov, raziskanostjo prostora ter hitrostjo raziskovanja najslabša izmed vseh strategij.



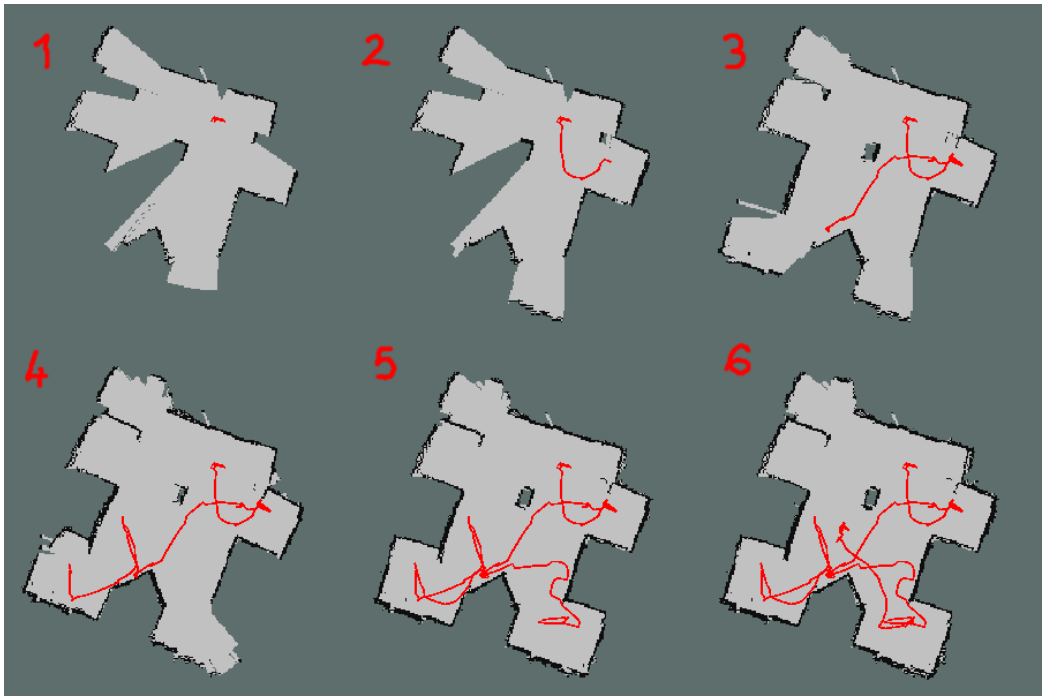
Slika 4.19: Potek raziskovanja strategije oddaljenosti in orientacije



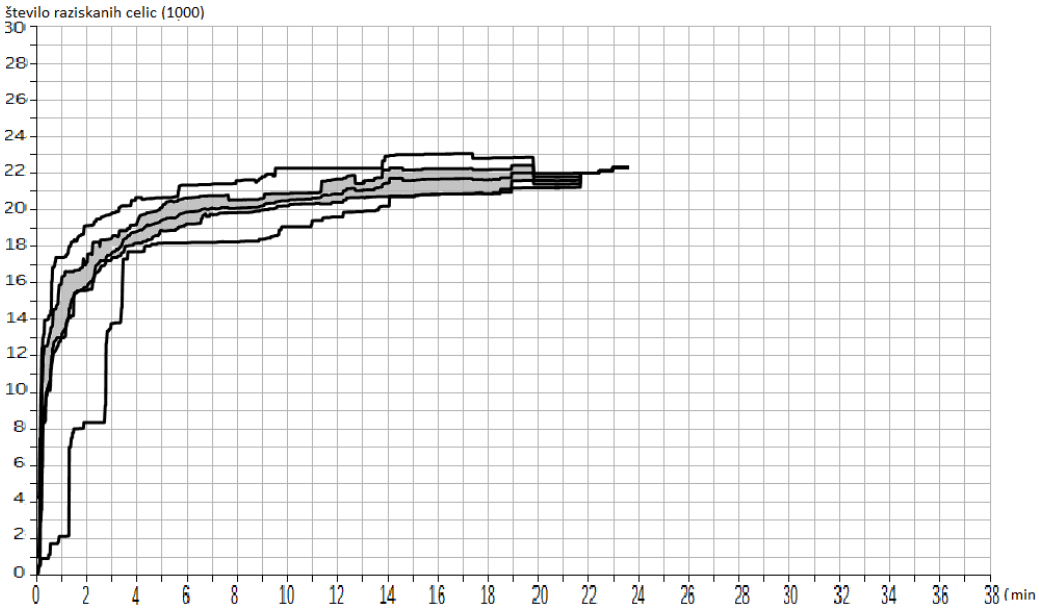
Slika 4.20: Graf postopka raziskovanja strategije oddaljenosti in orientacije

4.6.7 Strategija orientacije in pridobitve informacij

Zaradi vpliva ocen orientacije in pridobitve informacij smo za to strategijo pričakovali dolg in prepleten postopek raziskovanja, vendar pa je skupno upoštevanje ocen najboljših osnovnih strategij ustvarilo krajšo raziskovalno pot. Ko pa je razvidno s slike 4.21, se zaradi načina izbire raziskovalnih ciljev robot vseeno večkrat vrne v že raziskan del prostora, kar pa pozitivno vpliva na natančnost izdelanih zemljevidov, ki je primerljiva s naključno strategijo. Z izjemno dobrimi povprečji in nizkimi odstopanji pri času raziskovanja in raziskanosti prostora, ki presegajo večino ostalih strategij, uspešno združi najboljše vrednosti strategije orientacije in konsistentnost nizkih odstopanj od povprečnih vrednosti strategije pridobitve informacij. Kot lahko vidimo na sliki 4.22 ima s povprečno 3:18 minut do 18243 raziskanih celic zemljevida in konsistentno nizkim odklonom prav tako najboljši proces raziskovanja in je tako najboljša raziskovalna strategija.



Slika 4.21: Potek raziskovanja strategije orientacije in pridobitve informacij

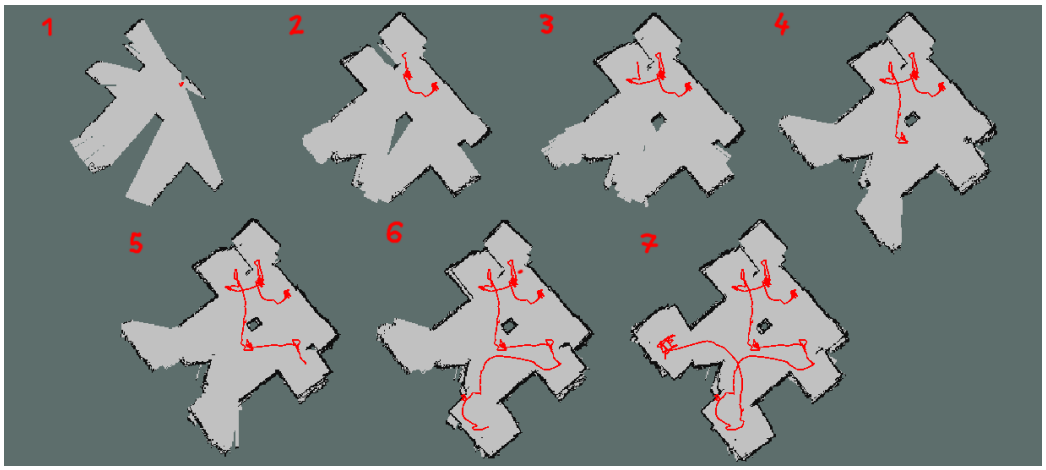


Slika 4.22: Graf postopka raziskovanja strategije orientacije in pridobitve informacij

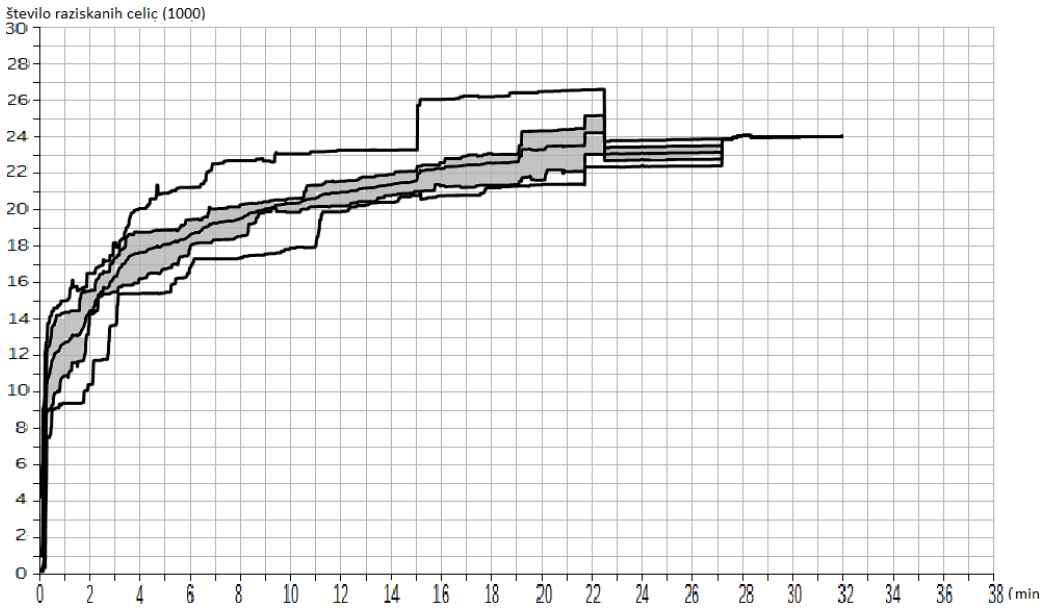
4.6.8 Strategija oddaljenosti, orientacije in pridobitve informacij

Zaradi vpliva ocen orientacije in pridobitve informacij ta strategija za raziskovalne cilje izbira robotu najbližja obrobja, ki so velika in imajo podobno orientacijo kot robot. Temu primerno je pot raziskovanja zaradi kriterija oddaljenosti večinoma konsistentna in usmerjena, vseeno pa vpliv kriterijev orientacije ter pridobitve informacij povzroča izbiro cilja in vračanje na že raziskano stran prostora. Primer takega raziskovanja lahko vidimo na sliki **??**. Medtem ko smo s kombinirano strategijo, ki enakomerno upošteva vse tri ocene raziskovanja, poskusili zmanjšati slabosti in obdržati prednosti, ki jih imajo osnovne strategije, pa se ta obnaša le kot povprečje slednjih. Tako ima v primerjavi z ostalimi slabši povprečni čas raziskovanja, raziskanost in natančnost ustvarjenih zemljevidov. Kot lahko razberemo s slike **??**, ki prikazuje graf raziskovanja ima v primerjavi s povprečjem strategij prav tako

počasnejši proces raziskovanja, saj povprečno doseže raziskanost 18243 ali 100% celic idealnega zemljevida v 5:21 minutah.



Slika 4.23: Potek raziskovanja strategije oddaljenosti, orientacije in pridobitve informacij



Slika 4.24: Graf postopka raziskovanja strategije oddaljenosti, orientacije in pridobitve informacij

Poglavje 5

Sklepne ugotovitve

V diplomski nalogi smo na mobilni platformi Turtlebot, ki ima nameščeno razvojno okolje ROS, implementirali in preizkušali avtonomno raziskovanje prostora, pri čemer smo primerjali učinkovitost strategij upoštevanja različnih ocen raziskovanja pri izbiri raziskovalnih ciljev. Dobljene rezultate lahko uporabimo za izboljšanje izbiranja ciljev in s tem učinkovitosti avtonomnega raziskovanja. Na ta način poskrbimo za boljšo porabo omejenih sredstev, kot sta električna energija in čas.

Izmed strategij izbire raziskovalnih ciljev je strategija, ki upošteva le oceno oddaljenosti robota od zastavljenega cilja, s počasnejšim začetnim raziskovanjem, nižjo mero raziskanosti in daljšim raziskovalnim časom prostora ter eno slabših natančnosti izdelanih zemljevidov v primerjavi s strategijo naključnega izbora cilja najslabša osnova strategija. Slab vpliv ocene oddaljenosti je opazen tudi pri kombiniranih strategijah, pri katerih sta poleg ocene oddaljenosti upoštevani tudi drugi oceni raziskovanja, ki blažita slabosti ocene oddaljenosti. Preostali osnovni strategiji, ki upoštevata le eno oceno, sta se obnesli dosti bolje. Medtem ko sta čas raziskovanja in raziskanost prostora povprečna, dosega osnovna strategija, ki upošteva zgolj oceno pridobitve informacij, najvišjo povprečno natančnost in ima najhitrejši proces raziskovanja. Podobno je strategija, ki upošteva oceno razlike v orientaciji, z relativno hitrim procesom raziskovanja, zelo nizkim časom

raziskovanja in visoko natančnostjo najboljša osnovna strategija. Medtem ko ti oceni v kombinacijah z ostalimi dosegata povprečne rezultate, prideta najbolj do izraza, ko se upoštevata samo ti dve hkrati. Ta kombinirana strategija enakomernega upoštevanja ocen pridobitve informacij in spremembe orientacije ima pri vseh kriterijih nekatere izmed najboljših povprečnih vrednosti s konsistentno najnižjimi odstopanji od povprečja in je tako najboljša strategija upoštevanja raziskovalnih ocen.

Na podlagi teh rezultatov smo ugotovili prednosti in slabosti strategij, kar nam omogoča izbiro najboljše strategije za naše razmere. Pričakovali smo, da se bodo različne strategije upoštevanja ocen raziskovanja pri izbiri raziskovalnih ciljev dobro izkazale pri nekaterih kriterijih in slabše pri drugih. Na ta način bi lahko glede na potrebe izbirali strategije, ki so specializirane za določen kriterij. Tako bi lahko v posameznem primeru, kot je kartiranje prostora za uporabo avtonomne kosilnice, ko potrebujemo visoko stopnjo natančnosti in raziskanosti prostora ter nam je vseeno za čas raziskovanja, ali pa v primeru iskanja ponesrečencev po potresu ali plazu, ko sta hitrost oziroma kratek raziskovalni čas in visoka raziskanost pomembnejša od natančnosti zemljevidov, izbrali določeno strategijo. Vendar pa smo na podlagi dobljenih rezultatov prišli do dveh ugotovitev. Prva je ta, da med različnimi strategijami ni opaziti večjih razlik v učinkovitosti, saj so povprečni rezultati ene strategije večinoma nahajajo znotraj mej odstopanja od povprečja drugih. Druga ugotovitev je, da ni strategije, ki bi lahko prinesla najboljše rezultate s vidika enega kriteriju za ceno slabih rezultatov pri drugemu, saj ima strategija, ki pri izbiri raziskovalnih ciljev oceno spremembe orientacije in oceno pridobitve informacij, konsistentno najboljše povprečne rezultate pri vseh kriterijih raziskovanja, zaradi česa lahko uporabo ostalih strategij zanemarimo .

Diplomsko nalogo bi se dalo izboljšati tako, da bi vzorec desetih uspešnih raziskovanj za posamezno strategijo povečali na dvajset ali več. Tako bi imeli več podatkov, s katerimi bi lahko zagotovili bolj reprezentativne in zanesljive povprečne vrednosti in odstopanja za primerjavo. Poleg tega bi lahko na-

mesto strategij, ko se izbrane strategije enakomerno upoštevajo, preizkušali tudi tiste, ki dajejo večjo pomembnost izbranim ocenam. Na ta način bi natančneje ugotovili, kako določena ocena vpliva na izbor ciljev in s tem na raziskovanje. Obe izboljšavi sta v primerjavi z enakomernim upoštevanjem in z uporabo trenutne metodologije desetih uspešnih raziskovanj prostora za vsako strategijo zamudnejši in zahtevnejši. Zato bi bila bolj učinkovita implementacija postopka raziskovanja v ROS-ovem simulacijskem okolju GAZEBO, ki pa v trenutni obliki zaradi napak pri razumevanju navodil premikanja robota deluje napačno in tako onemogoča pravilno preizkušanje postopkov in strategij raziskovanja. Medtem ko bi bilo testiranje s simulacijskim okoljem GAZEBO bolj učinkovito ter manj časovno zamudno, pa zaradi pomanjkanja realističnih testnih podatkov dobljeni rezultati ne bi imeli enake verodostojnosti kot ti, ki smo jih za potrebe diplomskega dela zbrali z izvajanjem eksperimentov s pravim robotom v pravem okolju.

Literatura

[1] Robot. [Online]. Dosegljivo: https://sl.wikipedia.org/wiki/Robot. [Dostopano 10. 8. 2015].

[2] Robot 2. [Online]. Dosegljivo: http://www.allonrobots.com/. [Dostopano 10. 8. 2015].

[3] Rover. [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Rover_(space_exploration) . [Dostopano 10. 8. 2015]

[4] Avtonomno podvodno vozilo. [Online]. Dosegljivo: https://www.whoi.edu/main/ABE. [Dostopano 10. 8. 2015].

[5] Autonomni robot. [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Autonomous_robot. [Dostopano 10. 8. 2015].

[6] Brian Yamauchi, "A frontier-based approach for autonomous exploration", *Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation*, str. 146–151, 1997.

[7] Christian Dornhege, Alexander Kleineri, "A frontier-void-based approach for autonomous exploration in 3d", *Proceedings of the IEEE International Symposium onSafety, Security, and Rescue Robotics*, str. 351 – 356, 2011.

[8] S. Thrun, S. Thayer, W. Whittaker, C. Baker, W. Burgard, D. Ferguson, D. Hahnel, M. Montemerlo, A. Morris, Z. Omohundro, C. Reverte, W. Whittaker, "Autonomous exploration and mapping of abandoned mines", *Robotics and Automation Magazine, IEEE*, št. 11, zv. 4, str. 79 – 91, 2004.

[9] ROS. [Online]. Dosegljivo: http://www.ros.org/about-ros/. [Dostopano 19. 9. 2014].

[10] ROS. [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Robot_Operating_System. [Dostopano 10. 8. 2015].

[11] ROS exploration stack. [Online]. Dosegljivo: http://wiki.ros.org/exploration. [Dostopano 10. 8. 2015].

[12] ROS globalni planner. [Online]. Dosegljivo: http://wiki.ros.org/global_planner. [Dostopano 10. 8. 2015].

[13] ROS local planner. [Online]. Dosegljivo: http://wiki.ros.org/base_local_planner. [Dostopano 10. 8. 2015].

[14] E. W. Dijkstra, "A note on two problems in connexion with graphs", *Numerische Mathematik*, št. 1, str. 269–271, 1959.

[15] ROS local planner. [Online]. Dosegljivo: http://wiki.ros.org/costmap_2d. [Dostopano 10. 8. 2015].

[16] Z. Wang, A. C. Bovik, H. R. Sheikh, E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity", *Transactions on Image Processing, IEEE*, št. 13, zv. 4, str. 600–612, 2004.

[17] P. J. Besel, N. D. McKay, "A Method for Registration of 3-D Shapes", *Transactions on Image Processing, IEEE*, št. 14, zv. 2, str. 600–612, 2004.

[18] ICP. [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Iterative_closest_point. [Dostopano 10. 8. 2015].

[19] I. T. Jolliffe, "Principal Component Analysis", 2002.

[20] PCA. [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Principal_component_analysis. [Dostopano 10. 8. 2015].

[21] Kinect. [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Kinect. [Dostopano 10. 8. 2015].

[22] Turtle robot. [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Turtle_(robot). [Dostopano 10. 8. 2015].

[23] Turtle robot. [Online]. Dosegljivo: http://roamerrobot.tumblr.com/post/23079345849/the-history-of-turtle-robots. [Dostopano 22.12. 2015].

[24] Turtlebot. [Online]. Dosegljivo: http://www.turtlebot.com/. [Dostopano 10. 8. 2015].

[25] Turtlebot. [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Turtlebot. [Dostopano 10. 8. 2015].

[26] iRobot Roomba. [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Roomba. [Dostopano 10. 8. 2015].

[27] Robotic lawn mowers. [Online]. Dosegljivo: https:http://www.husqvarna.com/us/products/robotic-lawn-mowers/. [Dostopano 15. 12. 2015].

[28] SLAM. [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Simultaneous_localization_and_mapping. [Dostopano 10. 8. 2015].

[29] M. Montemerlo, S. Thrun, D. Koller, B. Wegbreit, "FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem",*Proceedings of the AAAI National Conference on Artificial Intelligence*, 2002.

[30] PCL. [Online]. Dosegljivo: http://pointclouds.org/about/. [Dostopano 10. 8. 2015].