Planiranje gibanja autonomnih mobilnih robota u dinamičkim i nepoznatim unutarnjim prostorima

Đakulović, Marija

Sveučilište u Zagrebu
Fakultet Elektrotehnike i Računarstva

Marija Đakulović

Planiranje gibanja autonomnih mobilnih robota u dinamičkim i nepoznatim unutarnjim prostorima

Doktorska disertacija

Zagreb, 2010.

Doktorska disertacija je izrađena na Zavodu za automatiku i računalno inženjerstvo, Fakulteta elektrotehnike i računarstva u Zagrebu

Mentor: prof.dr.sc. Ivan Petrović

Disertacija ima 203 stranice.

Rad br.

Povjerenstvo za ocjenu doktorske disertacije:

1. Dr.sc Mato Baotić, docent
   Sveučilište u Zagrebu Fakultet elektrotehnike i računarstva
2. Dr.sc Ivan Petrović, redoviti profesor
   Sveučilište u Zagrebu Fakultet elektrotehnike i računarstva
3. Dr.sc Robert Manger, redoviti profesor
   Sveučilište u Zagrebu Prirodoslovno matematički fakultet

Povjerenstvo za obranu doktorske disertacije:

1. Dr.sc Mato Baotić, docent
   Sveučilište u Zagrebu Fakultet elektrotehnike i računarstva
2. Dr.sc Ivan Petrović, redoviti profesor
   Sveučilište u Zagrebu Fakultet elektrotehnike i računarstva
3. Dr.sc Robert Manger, redoviti profesor
   Sveučilište u Zagrebu Prirodoslovno matematički fakultet
4. Dr.sc Nedjeljko Perić, redoviti profesor
   Sveučilište u Zagrebu Fakultet elektrotehnike i računarstva
5. Dr.sc Mario-Osvin Pavčević, izvanredni profesor
   Sveučilište u Zagrebu Fakultet elektrotehnike i računarstva

Datum obrane disertacije: 28. svibnja 2010. godine

Zahvala

U područje mobilne robotike uveo me moj mentor prof.dr.sc Ivan Petrović, kojem bih željela izraziti najveću zahvalnost na svim znanstvenim savjetima te nesebičnoj pomoći i podršci tijekom svih godina mog istraživačkog rada.

Posebno bih željela zahvaliti doc.dr.sc. Jadranku Matušku na savjetima i velikoj pomoći u nastavi i znanstvenom istraživanju tijekom svih ovih godina. Također hvala doc.dr.sc. Mati Baotiću na iznimnoj pomoći i konstruktivnim savjetima u završnoj fazi stvaranja ovog rada.

Zahvalnost dugujem i bivšim studentima Petru Mostarcu, Ivanu Kašiću, Šandoru Ilešu i Borku Dimitrijeviću koji su svojim diplomskim radovima značajno pomogli u izradi ovog rada.

Zahvaljujem svim svojim kolegama na Zavodu, Vedrani, Andreji, Martini, Tamari, Nikolici, Srećku, Ivanu, Mišelu, Edi, Vedranu, Tomislavu, Toniju, Vlahu i Damiru na velikoj podršci i poticaju.

Najljepša hvala mojoj obitelji i prijateljima, a posebno roditeljima Ivančici i Alojziju, sestri Almi, bratu Lovri, noni Đurđici te mojoj boljoj polovici – suprugu Berislavu na pruženoj podršci i ljubavi u svim trenucima.

Marija Đakulović
Zagreb, rujan, 2010.

Sadržaj

Poglavlje 1

Uvod

Autonomni su mobilni roboti strojevi koji se autonomno gibaju i obavljaju razne korisne poslove u dinamičkim i nepoznatim prostorima bez njihova preinačavanja. Primjene autonomnih mobilnih robota vrlo su široke, primjerice prijenos tereta, čišćenje/pretraživanje prostora, nadzor prostora i razne druge uslužne primjene u uredima, skladištima, supermarketima, zračnim lukama, kućanstvima, bolnicama, muzejima itd. U navedenim se primjenama postavljaju vrlo složeni zahtjevi na sustav planiranja gibanja mobilnog robota, budući da se u istim prostorima mogu gibati ljudi i razni objekti. Osim toga, da bi se izbjegla potreba zadavanja CAD (engl. *computer-aided design*) modela prostora mobilnom robotu, sustav planiranja gibanja treba omogućiti i njegovo autonomno gibanje u nepoznatim prostorima. U ovoj se disertaciji istražuju upravo problemi planiranja gibanja mobilnih robota u takvim dinamičkim i nepoznatim prostorima s ciljem razvoja cjelovitog sustava planiranja gibanja koji će osigurati sigurno gibanje mobilnog robota kroz prostor bez sudara s drugim dinamičkim objektima u prostoru uz istodobno obavljanje postavljenog zadatka.

Istraživanjima su u disertaciji obuhvaćena četiri glavna istraživačka problema u planiranju gibanja mobilnog robota: (1) planiranje geometrijskih putanja robota u poznatim prostorima s ciljem razvoja algoritma koji daje optimalnu putanju i koji je prikladan za rad u stvarnome vremenu i pri promjenama u prostoru; (2) planiranje geometrijskih putanja robota u nepoznatim prostorima s ciljem razvoja algoritma za iscrpno istraživanje prostora radi izgradnje njegove karte; (3) generiranje trajektorije gibanja robota radi osiguranja dobrog slijeđenja geometrijske putanje robota, poštujući njegova kinematička i dinamička ograničenja i raspored prepreka u njegovoj okolini te (4) planiranje gibanja u prostorima s gibajućim preprekama s ciljem razvoja algoritma za sigurno izbjegavanje sudara s njima i algoritma za sprječavanje zastoja u uskim prolazima.

Većina algoritama planiranja geometrijskih putanja generira graf mogućih putanja od aktualne do ciljne pozicije robota na osnovi poznavanja karte prostora. Optimalna se putanja tada nalazi nekom od strategija pretraživanja grafova. Često se koristi tzv. D* algoritam koji je vrlo pogodna strategija pretraživanja u navigaciji mobilnih robota, jer omogućuje promjene putanje u stvarnome vremenu pri promjenama u prostoru. To

se svojstvo naziva dinamičkim planiranjem. Kao graf mogućih putanja često se koriste mrežaste karte zauzeća prostora koje prikazuju prostor mrežom kvadratnih polja jednakih površina. Svako polje sadrži informaciju o vjerojatnosti zauzeća toga dijela prostora i predstavlja čvor grafa mogućih putanja. Algoritmi planiranja putanje zasnovani na mrežastim kartama imaju dva značajna nedostatka: (1) dobiveni je put sastavljen od niza povezanih linijskih segmenata s promjenama u orijentaciji segmenata koje su višekratnici od 45° i (2) računski su previše zahtjevni za planiranje putanja u velikim prostorima. Prvi se nedostatak odražava u potrebi za znatnim usporavanjem (često i zaustavljanjem) gibanja mobilnoga robota zbog skokovitih promjena orijentacije pri prelasku s praćenja jednog linijskog segmenta putanje na drugi. S ciljem rješavanja toga problema u ovom se radu istražuje algoritam planiranja putanje kojemu orijentacije segmenata putanje nisu višekratnici 45° već su proizvoljne. Drugi se nedostatak rješava predstavljanjem prostora mrežastom kartom nejednolike razlučivosti kod koje veličine polja ovise o gustoći prepreka u prostoru ili uvođenjem hijerarhijske apstrakcije prikaza prostora. U ovom se radu razvija hijerarhijski algoritam planiranja putanje.

Problem planiranja geometrijske putanje dodatno se usložnjava ako se mobilni robot mora autonomno gibati u nepoznatome prostoru, tj. u prostoru za koji mu karta nije dostupna. Taj se problem često naziva problemom istraživanja prostora. Većina strategija istraživanja prostora gura robota na granicu istraženog i neistraženog prostora. Strategije većinom nisu pouzdane kada se primjene u realnim uvjetima zbog nesigurnosti mjerenja, lokalizacije i izgradnje karte. Druge pak strategije pate od velike računske složenosti otkrivanja granica između istražena i neistražena prostora i odabira najobećavajućije granice. Stoga se u ovome radu razvija algoritam istraživanja prostora koji uzima u obzir pogreške mjerenja senzora i nesigurnost položaja u izgradnji karte prostora.

Po dijelovima prekinuta geometrijska putanja pretvara se u neki oblik glatke krivulje koja u vremenu predstavlja trajektoriju mobilnog robota na osnovi koje se robotu zadaju referentne brzine gibanja. Razlikuju se neposredni i posredni pristupi generiranju trajektorije. U neposrednim se pristupima kod generiranja trajektorije moraju uzeti u obzir dinamička i kinematička ograničenja robota. Dobivena krivulja prostorno odstupa od izvorne geometrijske putanje, pa se mora dodatno provjeriti prolazi li ona kroz neku prepreku u prostoru. U slučaju dinamičkih prostora, reparametriranje globalne trajektorije takvim metodama računski je dosta zahtjevno. Posredni pristupi generiranju trajektorije zasnivaju se na primjeni algoritama izbjegavanja prepreka. Korištenje algoritma izbjegavanja prepreka u koordinaciji s globalnom geometrijskom putanjom neizbježno je u dinamičkim prostorima u kojima proračun upravljačkih naredaba mora biti dovoljno brz da se mobilni robot stigne prilagoditi promjenama u prostoru bez neželjenog zaustavljanja na putu do cilja. U ovome se radu istražuje posredni pristup generiranja trajektorija s ciljem razvoja algoritma planiranja gibanja koji objedinjuje dva ključna problema: planiranje putanje i slijeđenje putanje. Dodatno, razmatra se konvergencija gibanja robota prema zadanoj ciljnoj poziciji, pri čemu se koristi Ljapu-

novljeva analiza stabilnosti.

Veliki je izazov u planiranju gibanja mobilnih robota riješiti probleme izazvane prisutnošću gibajućih prepreka u prostoru. Dvije su glavne kategorije pristupa rješavanju toga problema: prostorno-vremenski pristupi i pristupi proširenjem potencijalnih polja. Prostorno-vremenski pristupi pretpostavljaju *a priori* znanje o trajektorijama pokretnih prepreka. Pristupi proširenjem potencijalnih polja generiraju umjetnu silu koja odbija robota od prepreka. Osim izbjegavanja sudara s gibajućim preprekama, potrebno je riješiti i problem zastoja u uskim prolazima. Zastoj se javlja u situacijama kada se robot i gibajuća prepreka (drugi robot, čovjek) sretnu u uskom prolazu i jedno drugom onemoguće nastavak gibanja prema cilju. Postoji nekoliko metoda koje problem rješavaju postavljanjem prioriteta robotima, ali njihova uspješnost nije zajamčena u svim situacijama. U ovome se radu razvija algoritam planiranja gibanja koji osigurava sigurno i glatko gibanje bez zastoja među statičkim i gibajućim preprekama.

Ostatak je ovog rada organiziran kako slijedi.

U **drugom** su poglavlju opisani algoritmi planiranja geometrijskih putanja robota u poznatim prostorima, ali s mogućnošću pojave nepoznatih statičkih i dinamičkih prepreka u njima. Budući da je naglasak na rad u stvarnom vremenu, detaljnije su opisani algoritam A* i njegovi nasljednici, algoritmi D* i FD*, koji imaju svojstvo dinamičkog planiranja. Nadalje, opisan je razvijeni dvosmjerni algoritam D* planiranja geometrijske putanje zasnovan na mrežastoj karti zauzeća, koji izračunava najkraću putanju u geometrijskom prostoru i nasljeđuje svojstvo dinamičkog planiranja od algoritma D*. Na kraju poglavlja dani su eksperimentalni rezultati provjere razvijenog algoritma planiranja putanje.

U **trećem** se poglavlju razmatra problem planiranja putanje u velikim unutarnjim prostorima, kao što su prostori sastavljeni od više katova ili čak više zgrada. Opisan je Cagigasov algoritam kojim je inspiriran razvijeni hijerarhijski algoritam planiranja putanje nazvan FHD*. Nadalje, opisana je nova hijerarhijska organizacija karte prostora kojom je osigurano optimalno horizontalno (u jednom katu) i vertikalno planiranje putanje (između katova zgrade). Prikazana je razvijena metoda automatske izgradnje takve karte iz mrežaste karte zauzeća. Na kraju poglavlja dani su eksperimentalni rezultati provjere razvijenog hijerarhijskog algoritma planiranja putanje.

U **četvrtom** se poglavlju razmatra istraživanje nepoznatog prostora s ciljem izgradnje karte prostora. Dan je opis Ekmanova algoritma istraživanja koji pretpostavlja idealnu lokalizaciju robota i idealni senzor udaljenosti. Opisan je algoritam izdvajanja linijskih segmenata iz zašumljenih očitanja senzora, uzimajući pritom u obzir utjecaj nesigurnosti položaja mobilnog robota. Nadalje, opisan je razvijeni algoritam istraživanja prostora koji uklanja stroga ograničenja Ekmanova algoritma na senzor udaljenosti i lokalizaciju robota. Na kraju poglavlja dani su simulacijski i eksperimentalni rezultati provjere razvijenog algoritma istraživanja prostora.

U **petom** su poglavlju opisani algoritmi slijeđenja putanje zasnovani na algoritmu dinamičkog prozora koji proračunavaju moguće trajektorije s obzirom na kinematička

i dinamička ograničenja robota. Opisan je razvijeni algoritam planiranja gibanja koji objedinjuje algoritam planiranja putanje i algoritam dinamičkog prozora. Razvijenim objedinjenim algoritmom planiranja gibanja jamči se sigurno gibanje robota do globalnog cilja u prisustvu statičkih prepreka. Zatim je opisano proširenje razvijenog algoritma planiranja gibanja koje osigurava sigurno i glatko gibanje bez zastoja i među gibajućim preprekama. Na kraju poglavlja dani su simulacijski i eksperimentalni rezultati provjere razvijenog algoritma planiranja gibanja.

U **šestom** je poglavlju opisan integrirani algoritam planiranja gibanja mobilnog robota koji osigurava konvergenciju gibanja robota prema zadanoj ciljnoj poziciji. Razvijeni se algoritam planiranja gibanja zasniva na konceptu pomičnog horizonta. Za razvijeni je algoritam planiranja gibanja dokazana asimptotska stabilnost ciljne točke korištenjem Ljapunovljeve analize stabilnosti. Zatim su dani simulacijski rezultati provjere razvijenog algoritma planiranja gibanja.

U **sedmom** su poglavlju dana zaključna razmatranja problema planiranja gibanja mobilnog robota u dinamičkim i nepoznatim unutarnjim prostorima.

Poglavlje 2

Planiranje putanje u poznatim dinamičkim prostorima

Putanja mobilnog robota definirana je kao uređeni skup svih konfiguracija koje mobilni robot poprima na putu od početne konfiguracije u kojoj se nalazi do ciljne konfiguracije u koju treba doći. Pojam konfiguracijski prostor uveden je u robotiku krajem 70-ih godina prošlog stoljeća [75], a označava sve moguće konfiguracije robota u geometrijskom prostoru. Konfiguracija ili stanje robota u geometrijskom prostoru određeno je njegovim položajem koji označava poziciju i orijentaciju robota u koordinatnom sustavu prostora.

U skladu s navedenim definicijama putanje i konfiguracijskog prostora definiran je i problem planiranja putanje kao problem odabira optimalnog skupa konfiguracija mobilnog robota od početne do ciljne konfiguracije iz skupa svih mogućih konfiguracija sadržanih u konfiguracijskom prostoru. Prema [68] pristupi planiranja putanje mogu se podijeliti u tri glavne skupine: potencijalna polja (engl. *potential field*), karta putanja (engl. *roadmap*) i dekompozicija prostora (engl. *cell decomposition*).

Potencijalna polja proizvode silu koja odbija robota od prepreka i privlači ga k cilju [59]. Taj pristup izravno određuje gibanje robota do cilja, ali ima problem lokalnog minimuma. Postoje modifikacije potencijalnih polja koje uklanjaju lokalni minimum, ali takvi su pristupi računski prezahtjevni [105].

Karta putanja predstavlja povezanost konfiguracijskog prostora u obliku grafa putanja u poligonalnom prikazu prostora [132], [135], [12]. Putanja od starta do cilja pronalazi se pretraživanjem toga grafa. Nedostatak je toga pristupa što se često teško osigurava svojstvo kompletnosti, to jest ne jamči se pronalazak putanje do cilja iako ona postoji [132]. Drugi je nedostatak u primjeni ovoga pristupa na robotu s ograničenim dometom senzora udaljenosti, jer u tom slučaju mogu nedostajati potrebne informacije o prostoru da se izgradi karta putanja [136].

Dekompozicija prostora dijeli konfiguracijski prostor na segmente prostora i tada nalazi slijed tih segmenata koji čine putanju [68]. Svaki segment prostora predstavlja čvor u grafu koji se pretražuje. Dekompozicija prostora može biti egzaktna ili aproksimativna. Dok egzaktna dekompozicija egzaktno opisuje čitavi konfiguracijski prostor,

aproksimativna je dekompozicija računski manje zahtjevna pa se češće primjenjuje. Aproksimativna se dekompozicija najčešće zasniva na segmentima kvadratnih oblika. Primjer je takve dekompozicije mrežasta karta zauzeća [130] koja se koristi u radu. Brojni su algoritmi pretraživanja grafa primijenjeni na takvoj dekompoziciji prostora [126], [127], [64], [31], koji omogućuju brzo preračunavanje putanje u slučaju promjena u prostoru. Nedostatak je ovoga pristupa da također ne osigurava kompletnost, ali samo u slučaju kada je veličina kvadratnih segmenata prevelika tako da se može dogoditi da robot mora proći kroz prolaz koji je uži od stranice kvadratnog segmetna. Prema tome, kompletnost se može zajamčiti smanjenjem veličine kvadratnih segmenata, što naravno povećava računsku složenost algoritama planiranja putanje, jer se povećava broj čvorova grafa koje treba pretražiti. Smanjenje računske složenosti može se postići primjenom hijerarhijske reorganizacije prostora, što je opisano u 3. poglavlju.

U ovom se poglavlju razmatra problem planiranja najkraće putanje slobodne od prepreka od početne do ciljne pozicije robota uz pretpostavku da je veličina prepreka upisana u kartu prostora. Pretpostavlja se da je mobilni robot opremljen senzorom udaljenosti ograničenog dometa kojim otkriva prepreke u prostoru i nadograđuje kartu. Koristi se pristup planiranja putanje zasnovan na mrežastim kartama zauzeća i razmatraju se algoritmi pretraživanja prikladni za rad u stvarnom vremenu i pri promjenama u prostoru. Poglavlje je sadržajno organizirano kako slijedi. U potpoglavlju 2.1 dane su osnovne definicije procesa pretraživanja prostora stanja, a u potpoglavlju 2.2 definicije grafa stvorenog iz mrežaste karte zauzeća. U potpoglavlju 2.3 detaljnije su opisani i uspoređeni algoritmi pretraživanja takvog grafa koji su glavno polazište razvijenog algoritma koji je opisan u potpoglavlju 2.4. Zaključna razmatranja dana su u potpoglavlju 2.5.

2.1 Definicija procesa pretraživanja prostora stanja

U terminologiji područja umjetne inteligencije izraz planiranje odnosi se na pretraživanje diskretnog prostora stanja [107]. Nadalje, definira se konačan broj akcija koje se mogu primijeniti nad diskretnim skupom stanja i konstruira se rješenje zadavajem odgovarajućeg slijeda akcija. Proces pretraživanja definiraju sljedeće komponente:

- **početno stanje** u kojem se robot nalazi;
- **operator** koji opisuje izvršenu akciju u obliku $o(x) = x'$, odnosno daje stanje x' koje će se dohvatiti izvršavanjem neke akcije u stanju x. Alternativno se koristi i funkcija *sljednika* S, koja za dano stanje x, daje skup svih stanja koja se mogu dohvatiti jednostrukim akcijama (ne niz akcija nego jedna) iz stanja x.
- **prostor stanja** je skup svih stanja dohvatljivih iz početnog stanja slijedom izvršenih akcija.

- **putanja** u prostoru stanja definirana je kao bilo koji niz akcija provedenih od jednog stanja do drugog.
- **ciljni test** je funkcija prepoznavanja cilja i ona se primjenjuje u svakom stanju da bi se detektiralo ostvarenje cilja.
- **cijena putanje** je funkcija koja pridružuje brojčanu vrijednost putanji. U najčešćim slučajevima je cijena putanje zbroj svih pojedinačnih vrijednosti plaćenih za izvršavanje akcija duž putanje. Cijena putanje predstavlja kriterij po kojem bi rješenje problema bilo optimalno. Kriterij primjerice može biti proteklo vrijeme, prevaljena udaljenost ili utrošena energija. Označava se funkcijom g.

Rješenje procesa pretraživanja je putanja od početnog do ciljnog stanja. Do rješenja se dolazi pretraživanjem prostora stanja. Primjenjivanjem funkcije sljednika na trenutno stanje dolazimo do novog skupa stanja. Taj se proces naziva *proširivanjem stanja*, odnosno, određivanjem svih sljednika trenutnog stanja. U procesu pretraživanja izdvaja se jedna mogućnost iznad ostalih, koje se ostavljaju za kasnije razmatranje, u slučaju da izdvojena nije dovela do rješenja. Taj je odabir određen strategijom pretraživanja. Proces pretraživanja najbolje je prikazati grafom u kojem čvorovi predstavljaju stanja, a bridovi prijelaze stanja. Algoritam koji se primjenjuje naziva se *algoritmom pretraživanja grafa*. Određena se strategija pretraživanja ocjenjuje kroz četiri kriterija:

- **kompletnost** - osigurava li strategija pronalazak rješenja ako ono postoji;
- **vremenska složenost** - koliko traje pronalaženje rješenja;
- **memorijska složenost** - koliko je memorije potrebno za izvršavanje pretraživanja;
- **optimalnost** - pronalazi li strategija najbolje rješenje iz skupa mogućih rješenja po kriteriju cijene putanje.

Strategije pretraživanja mogu se podijeliti na *neinformiranu* i *informiranu* skupinu strategija. Neinformirane strategije nemaju nikakvu informaciju o broju koraka ili cijeni putanje od trenutnog stanja do cilja. Primjeri neinformiranih strategija su pretraživanje u širinu (BFS) (engl. *Breadth-first-search*) [107], Witkowskijev algoritam [134], uniformno pretraživanje ili Dijkstrin algoritam [25] i algoritam D* [126]. Informirane strategije sadrže te dodatne informacije i dolaze do rješenja mnogo učinkovitije. Informacije koje koristi informirana strategija pretraživanja koriste se u *funkciji ocjenjivanja* koja pridružuje ocjenu svakom čvoru koji se može proširiti čvorovima sljednicima. Čvor koji ima najbolju ocjenu bit će proširen prvi. Te se strategije nazivaju još i strategijama *pretraži-prvo-najboljeg* (engl. *best-first-search*). Primjeri informiranih strategija su pohlepno pretraživanje (engl. *Greedy-search*) [107], algoritam A* [48] i fokusirani algoritam D* (engl. *focussed D* algorithm*) [127]. U nastavku će biti detaljnije opisane neke od ovih strategija pretraživanja.

2.2 Prostor mobilnog robota kao graf

2.2.1 Skupovi i liste

U ovom su dijelu definirani pojmovi koji se koriste kroz cijeli rad [67].

Skup je neporedani niz različitih elemenata. Primjerice, zapisuje se kao $\mathcal{S} = \{1, 5, 8\}$ i vrijedi $\{5, 1, 8\} = \{1, 5, 8\}$. Pišemo $s \in \mathcal{S}$ kada želimo reći da je s element iz skupa S. *Kardinalnost* (konačnog) skupa $\mathcal{S}$, označena s $|\mathcal{S}|$, je broj elemenata u skupu $\mathcal{S}$. Primjerice, $|\{1, 5, 8\}| = 3$. Za nenegativni cijeli broj k, *k-skup* je skup kardinalnosti k. *Prazan skup* je skup koji nema elemenata. On je 0-*skup* i označava se s $\emptyset$. Za skupove $\mathcal{S}$ i $\mathcal{T}$ kažemo da je $\mathcal{S}$ *podskup* od $\mathcal{T}$ ako je svaki element iz $\mathcal{S}$ element iz $\mathcal{T}$. To je ekvivalentno uvjetu $s \in \mathcal{S} \Rightarrow s \in \mathcal{T}$ i pišemo $\mathcal{S} \subseteq \mathcal{T}$. Kažemo da je $\mathcal{S}$ *pravi podskup* od $\mathcal{T}$ ako vrijedi $\mathcal{S} \subseteq \mathcal{T}$ i $\mathcal{S} \neq \mathcal{T}$ i pišemo $\mathcal{S} \subset \mathcal{T}$. Presjek, unija i razlika skupova $\mathcal{S}$ i $\mathcal{T}$ definirani su redom

$$\begin{aligned} \mathcal{S} \cap \mathcal{T} &= \{s \mid s \in \mathcal{S} \text{ i } s \in \mathcal{T}\}, \\ \mathcal{S} \cup \mathcal{T} &= \{s \mid s \in \mathcal{S} \text{ ili } s \in \mathcal{T}\}, \\ \mathcal{S} \setminus \mathcal{T} &= \{s \mid s \in \mathcal{S} \text{ i } s \notin \mathcal{T}\}. \end{aligned}$$

Lista je poredani niz elemenata. Primjerice, $\mathcal{L} = [8, 1, 1]$ i vrijedi $[8, 1, 1] \neq [1, 8, 1]$. Elementi liste $\mathcal{L}$ označeni su s $\mathcal{L}[1], \ldots, \mathcal{L}[|\mathcal{L}|]$, redom, gdje je $|\mathcal{L}|$ *duljina* (broj elemenata) liste $\mathcal{L}$. U radu koristimo operaciju unije dviju lista tako što se liste nadovežu jedna na drugu. Neka su dane dvije liste $\mathcal{L}$ i $\mathcal{M}$. Unija dviju lista $\mathcal{L} \cup \mathcal{M}$ čini listu kojoj je lista $\mathcal{M}$ nadovezana na listu $\mathcal{L}$. Pritom je bitan poredak lista u operaciji, odnosno ne vrijedi komutativnost operacije. U slučaju da je zadnji element prve liste jednak prvom elementu druge liste, na mjestu nadovezivanja izostavlja se isti element. Prema navedenom unija lista definirana je kao

$$\mathcal{L} \cup \mathcal{M} = [a_1, a_2, \ldots, a_n],$$

gdje je

$$a_k = \begin{cases} \mathcal{L}[k] & \text{za } k = 1, \ldots, |\mathcal{L}| \\ \mathcal{M}[k - |\mathcal{L}|] & \text{za } \mathcal{M}[1] \neq \mathcal{L}[|\mathcal{L}|] \text{ i } k = |\mathcal{L}| + 1, \ldots, |\mathcal{L}| + |\mathcal{M}| \\ \mathcal{M}[k - |\mathcal{L}| + 1] & \text{za } \mathcal{M}[1] = \mathcal{L}[|\mathcal{L}|] \text{ i } k = |\mathcal{L}| + 1, \ldots, |\mathcal{L}| + |\mathcal{M}| - 1. \end{cases}$$

U radu također koristimo i oduzimanje elementa iz liste tako da se svi elementi liste nakon oduzetog elementa pomiču za jedno mjesto u lijevo. Neka je dana lista $\mathcal{L} = [a_1, \ldots, a_{k-1}, a_k, a_{k+1}, \ldots, a_n]$. Oduzimanje elementa liste a_k definirano je kao:

$$\mathcal{L} \setminus a_k = [a_1, \ldots, a_{k-1}, a_{k-2}, \ldots, a_n].$$

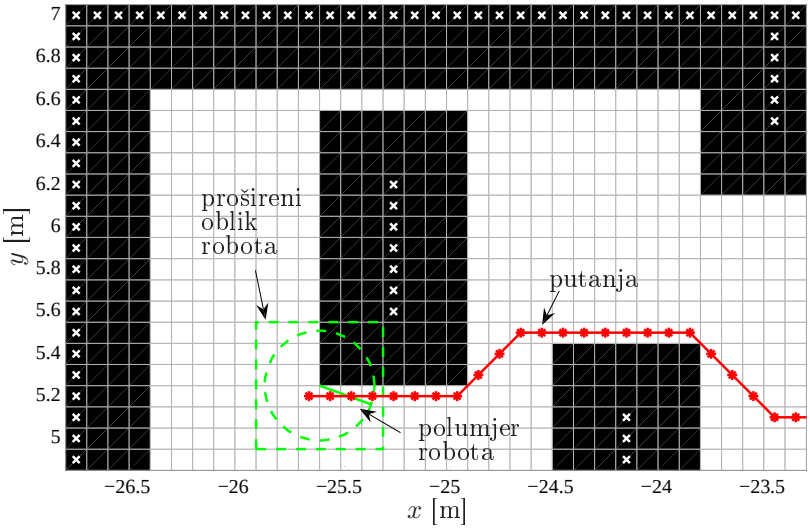
2.2.2 Mrežaste karte zauzeća

Većina algoritama planiranja putanje koristi mrežastu kartu zauzeća [130] kao graf pretraživanja. Karta je dobivena dijeljenjem prostora mrežom na K kvadratnih polja jednakih duljina stranica e_{polje}, koji su apstraktno prikazani skupom polja $\mathcal{K} = \{1, \ldots, K\}$ s odgovarajućim Kartezijevim koordinatama središta polja $c_i \in \mathbb{R}^2$, $i \in \mathcal{K}$. Svako polje sadrži informaciju o vjerojatnosti zauzeća toga dijela prostora. Binarne mrežaste karte zauzeća su one karte koje se sastoje samo od slobodnih i zauzetih polja. U takvim se kartama koristi binarna funkcija zauzeća $z(i) \in \{1, \infty\}$, $i \in \mathcal{K}$, kojom se opisuje skup svih prepreka u prostoru označenih skupom $\mathcal{Z} = \{i \in \mathcal{K} \mid z(i) = \infty\}$, a slobodna polja su sva ostala označena skupom $\mathcal{N} = \mathcal{K} \setminus \mathcal{Z}$.

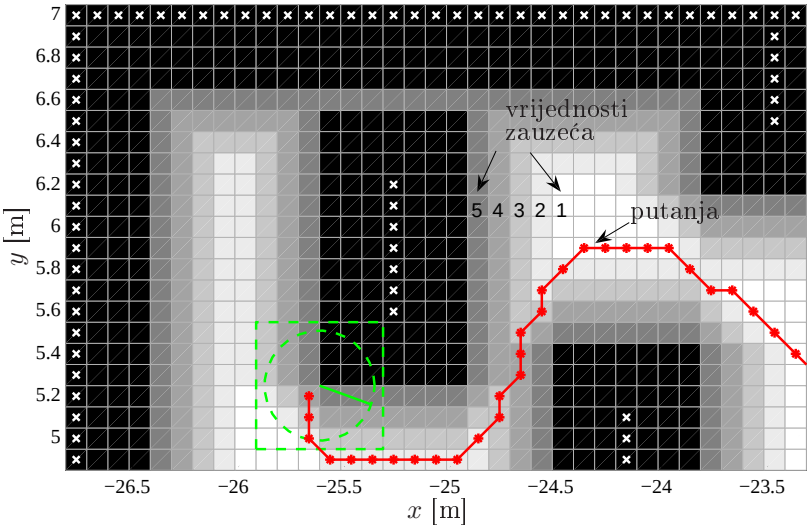
Robot se u takvoj karti predstavlja kvadratnom maskom, pri tom kvadrat mora biti takvih dimenzija da se opseg robota može upisati u njega. Za polovicu stranice kvadrata proširuju se prepreke te se robot u mrežastoj karti zauzeća smatra točkom. Kvadratna maska se uzima zato da se robot u svakoj točki slobodnog polja mrežaste karte zauzeća može okretati u mjestu, čime je olakšano planiranje putanje, odnosno potrebno je planirati samo pozicije točke u karti. Problem ovakvog pristupa je da se uski prolazi u karti mogu zatvoriti iz dva razloga (1) zbog preniske razlučivosti mreže, što se može riješiti smanjenjem veličine polja mreže tj. povećanjem razlučivosti mreže i (2) prolaz je preuzak da bi robot mogao rotirati u njemu, ali zapravo može proći kroz njega uz određenu orijentaciju robota u čijem slučaju kvadratni opis robota nije zadovoljavajuć. Za rješavanje ovog problema, koji se naziva problemom pomicanja klavira (engl. *piano mover's problem*) [69] nužno je koristiti odgovarajući opis robota (najčešće pravokutnik ili elipsu) te u problem planiranja putanje uključi osim planiranja pozicija i planiranje orijentacija na putanji.

U ovom radu koristimo mobilnog robota koji se u karti prostora dobro može opisati krugom polumjera r_r. Prepreke su proširene za cjelobrojni broj polja $\lceil r_r / e_{polje} \rceil$, odnosno robota opisujemo kvadratnom maskom stranice $2 \cdot \lceil r_r / e_{polje} \rceil \cdot e_{polje}$. Proširenja prepreka izvode se pridruživanjem središta kvadratne maske robota svakoj točki unutar polja sa stvarnom preprekom te postavljanjem zauzetima svih polja pod maskom robota. Ova procedura proširenja prepreka implementacijski je jednostavna i omogućuje jednostavnu detekciju sudara trajektorije robota s preprekom. Na slici 2.1 prikazan je opseg korištenog robota i njegovo proširenje kvadratnim oblikom u mrežastoj karti zauzeća. Može se primijetiti da su uglovi prepreka prošireni $\sqrt{2}$ puta više nego što je nužno, što dodatno potvrđuje sigurnost trajektorija koje ne prolaze kroz zauzeta polja.

U radu koristimo dva tipa mrežastih karata zauzeća: binarnu mrežastu kartu zauzeća i mrežastu kartu zauzeća sa *sigurnosnom maskom cijena* [115]. Sigurnosna maska cijena slična je polju umjetnih potencijala [59] široko korištenih u robotici. Uključuje se u mrežastu kartu zauzeća definiranjem novih vrijednosti funkcije zauzeća. Svako polje blizu prepreke unutar širine sigurnosne maske cijena M_c, koja je izražena u broju polja, dobiva određenu vrijednost zauzeća koja ovisi o udaljenosti polja od najbliže pre-



Slika 2.1. *Dio prostora predstavljen binarnom mrežastom kartom zauzeća: slobodna polja označena su bijelom bojom ($z(\cdot) = 1$), a zauzeta polja crnom bojom ($z(\cdot) = \infty$), gdje su sa x označene stvarne pozicije prepreka. Primjer određene putanje u takvoj karti označena je sa *. Polumjer robota iznosi $r_r = 0.26$ m, a odabrana veličina polja iznosi $e_{polje} = 0.1$ m.*



Slika 2.2. *Dio prostora predstavljen mrežastom kartom zauzeća sa sigurnosnom maskom cijena širine $M_c = 4$: slobodna polja označena su bijelom bojom ($z(\cdot) = 1$) i nijansama sive boje ($z(\cdot) \in \{2, 3, 4, 5\}$), a zauzeta polja crnom bojom ($z(\cdot) = \infty$), gdje su sa x označene stvarne pozicije prepreka. Primjer određene putanje u takvoj karti označena je sa *.*

preke. Najdalje polje unutar sigurnosne maske cijena dobiva vrijednost za jedan veću od vrijednosti slobodnog polja izvan sigurnosne maske cijena, a vrijednost polja unutar maske inkrementalno raste od najdaljeg polja prema zauzetom polju. Funkcija zauzeća poprima sljedeće vrijednosti:

$$z(i) = \begin{cases} \max\{1, (M_c + 2 - \min_j \|c_i - c_j\|_\infty)\} & \text{ako } i \in \mathcal{K} \setminus \mathcal{Z}, j \in \mathcal{Z} \\ \infty & \text{ako } i \in \mathcal{Z} \end{cases}, \tag{2.1}$$

gdje $\|\cdot\|_\infty$ označava *normu beskonačno*[1]. Na taj se način dobivaju gradijenti sigurnosnih vrijednosti od prepreka prema slobodnom prostoru. Prednost ovih gradijenata sigurnosnih vrijednosti jest da odmiču putanju dalje od prepreka radi sigurnosti gibanja, a pri tom i dalje omogućuju prolaz kroz uske hodnike što je bolje nego dodatno proširiti prepreke u čijem slučaju bi se uski prolazi zatvorili. Širina sigurnosne maske cijena, odnosno parametar M_c, utječe na odmaknutost putanje od prepreka i može se odrediti kao što je opisano u poglavlju 5.3.3. Slika 2.2 predstavlja mrežastu kartu zauzeća sa sigurnosnom maskom cijena dijela prostora, za parametar $M_c = 4$.

2.2.3 Definicije grafa

Graf $\mathcal{G}$ sastoji se od skupa elemenata $\mathcal{N}$ zvanih čvorovima i skupa elemenata $\mathcal{E}$ zvanih bridovima, takav da je svaki element od $\mathcal{E}$ dvočlani podskup skupa čvorova. Graf u kojem je dodatno definirana funkcija težine $w : \mathcal{E} \to \mathbb{R}$ naziva se težinskim grafom.

Traženje optimalne putanje izvodi se na težinskom grafu $\mathcal{G}(\mathcal{N}, \mathcal{E}, \mathcal{W})$ stvorenom iz mrežaste karte zauzeća. Slobodna polja predstavljaju skup čvorova $\mathcal{N} = \mathcal{K} \setminus \mathcal{Z}$ s odgovarajućim Kartezijevim koordinatama središta polja $c_i \in \mathbb{R}^2$, $i \in \mathcal{N}$. Kažemo da su dva polja c_i i c_j *susjedi* ako vrijedi $\|c_i - c_j\|_\infty = e_{polje}$. Skup bridova definiran je kao $\mathcal{E} = \{e_{i,j} := \{i, j\} \mid i, j \in \mathcal{N}, i \text{ i } j \text{ su susjedi}\}$. Skup težina bridova $\mathcal{W} = \{w_{i,j} \mid i, j \in \mathcal{N}, i \text{ i } j \text{ su susjedi}\}$ definiran je kao cijena prijelaza između dvaju susjednih polja

$$w_{i,j} := \|c_i - c_j\| \cdot \max\{z(i), z(j)\}. \tag{2.2}$$

U binarnim mrežastim kartama zauzeća razlikujemo dvije vrijednosti cijena prijelaza: ravni i dijagonalni prijelazi. U mrežastim kartama zauzeća sa sigurnosnom maskom cijena prijelazi su dodatno otežani s obzirom na blizinu prepreci. Graf u kojem postoji $\delta > 0$ takav da su sve težine u grafu veće od δ naziva se δ graf. Prema tome, razmatrani graf je δ graf, u kome je $\delta = e_{polje}$.

Kažemo da je lista $\mathcal{P} = \mathcal{P}(S, G)$ *putanja* od startnog čvora S do ciljnog čvora G u

[1] Za $x \in \mathbb{R}^n$ norma beskonačno definirana je kao $\|x\|_\infty := \max_{i \in \{1,\ldots,n\}} |x_i|$.

grafu $\mathcal{G}(\mathcal{N}, \mathcal{E}, \mathcal{W})$ ako:

$$\begin{aligned} \mathcal{P}[1] = S, \quad \mathcal{P}[|\mathcal{P}|] = G, \\ \mathcal{P}[i] \in \mathcal{N}, \quad i = 1, \ldots, |\mathcal{P}|, \\ \{\mathcal{P}[j], \mathcal{P}[j+1]\} \in \mathcal{E}, \quad j = 1, \ldots, |\mathcal{P}| - 1. \end{aligned} \tag{2.3}$$

Cijena putanje $\mathcal{P}$ definirana je kao zbroj težina bridova na putanji, odnosno,

$$c(\mathcal{P}) := \sum_{i=1}^{|\mathcal{P}|-1} w_{\mathcal{P}[i],\mathcal{P}[i+1]}. \tag{2.4}$$

Neka je $\pi(S, G)$ skup svih putanja od S do G u grafu $\mathcal{G}(\mathcal{N}, \mathcal{E}, \mathcal{W})$. *Optimalna cijena* putanje od S do G definirana je kao

$$\begin{aligned} g^*(S, G) = \quad & \min_{\mathcal{P}} \quad c(\mathcal{P}) \\ & \text{u.u.} \quad \mathcal{P} \in \pi(S, G) \end{aligned} \tag{2.5}$$

uz pretpostavku da je $g^*(S, G) = \infty$ ako je $\pi(S, G) = \emptyset$.

2.3 Pregled algoritama planiranja putanje u grafu

Neka je specificiran jedan startni čvor S i jedan ciljni čvor G u grafu $\mathcal{G}(\mathcal{N}, \mathcal{E}, \mathcal{W})$. Algoritam pretraživanja grafa nazivamo *dopustivim* ako jamči pronalazak optimalne putanje od S do G (kompletan i optimalan). Dopustivi algoritam pretraživanja grafa u danom grafu $\mathcal{G}(\mathcal{N}, \mathcal{E}, \mathcal{W})$, pronalazi: (1) optimalnu cijenu $g^*(S, G)$, i (2) jednu (može ih biti više) *optimalnu putanju* $\mathcal{P}^*(S, G)$ koja ima cijenu $g^*(S, G)$

$$\mathcal{P}^*(S, G) \in \{\mathcal{P} \in \pi(S, G) \mid g(\mathcal{P}) = g^*(S, G)\}, \tag{2.6}$$

ako postoji putanja za dani par čvorova $S, G \in \mathcal{N}$.

Radi jednostavnosti, simbolom $g^*(n)$ označit ćemo jedinstvenu cijenu optimalne putanje od čvora n do čvora G, odnosno $g^*(n) \equiv g^*(n, G)$. Simbolom $h^*(n)$ označit ćemo jedinstvenu cijenu optimalne putanje od čvora S do čvora n, odnosno $h^*(n) \equiv g^*(S, n)$.

Algoritam može pamtiti sve optimalne putanje od početnog čvora pretraživanja p (za neke algoritme je $p = S$, a za neke $p = G$) do čvora n. Proširivanjem čvorova postavljaju se *pokazivači* na prethodne čvorove koji imaju najmanju cijenu. Pokazivač se zapisuje preko funkcije $b(\cdot)$, gdje $b(n) = m$ označava da je čvoru n ostvarena najniža cijena preko čvora m. Po završetku algoritma, može se izdvojiti bilo koja putanja od čvora n, koji je bio proširen u procesu pretraživanja, do čvora p, od kojeg je počelo

pretraživanje, prema funkciji $b(\cdot)$ kako slijedi

$$\begin{aligned} \mathcal{P}^*[1] &= n, \\ \mathcal{P}^*[i] &= b(\mathcal{P}^*[i-1]), \quad i = 2, \ldots, |\mathcal{P}^*|. \end{aligned} \tag{2.7}$$

Proces izdvajanja putanje $\mathcal{P}(n, p)$ završava kada se dosegne početni čvor p, odnosno $\mathcal{P}^*[|\mathcal{P}^*|] = p$.

Razni dopustivi algoritmi razlikuju se u redoslijedu i broju pretraženih čvorova. Da bi algoritam proširio što je moguće manje čvorova u traženju optimalne putanje, mora stalno donositi što informiraniju odluku o tome koji čvor treba proširiti sljedeći. Primjerice, algoritam može proširivati nepotrebno one čvorove koji sigurno ne vode do cilja optimalnom putanjom ili, nasuprot tomu, može ignorirati one čvorove koji mogu biti na optimalnoj putanji i tako ne pronaći optimalnu putanju.

2.3.1 Algoritam A*

Algoritam A* koristi evaluacijsku funkciju f koju računa za svaki čvor n koji se sljedeći može proširiti i proširuje onaj čvor s najmanjom vrijednosti f. Vrijednost $f(n)$ predstavlja procjenu cijene optimalne putanje koja prolazi čvorom n. Sastavljena je kao

$$f(n) = h(n) + g(n), \tag{2.8}$$

gdje je $h(n)$ procjena cijene $h^*(n)$ (optimalne cijene od S do n), a $g(n)$ procjena cijene $g^*(n)$ (optimalne cijene od n do G). Procjenu $h(n)$ računa kao najmanju cijenu putanje od S do n proračunatu do trenutka proširivanja čvora n, odnosno vrijedi $h(n) \geq h^*(n)$. Kao procjenu g koristi bilo koju procjenu cijene optimalne putanje od n do G, dobivenu iz stvarnog problema koji je modeliran grafom, tzv. heurističnu funkciju ($g(n) = heuristika(n, G)$).

Heuristična funkcija mora biti *dopustiva*, odnosno ne smije precijeniti optimalnu putanju od čvora n do G ($g(n) \leq g^*(n)$). Ako je heuristika dopustiva tada je algoritam A* dopustiv (jamči nalaženje optimalne putanje) [48]. Dodatno, mora imati svojstvo *konzistentnosti*. To znači da bilo koja procjena $g(n)$ računata za čvor n prema fizikalnim informacijama stvarnog problema, ne smije biti poboljšana koristeći odgovarajuće podatke iz stvarnog problema za druge čvorove. Drugim riječima, heuristika ima svojstvo konzistentnosti ako vrijedi nejednakost trokuta:

$$heuristika(n, g) \leq g^*(n, m) + heuristika(m, g), \tag{2.9}$$

za $n, m, g \in \mathcal{N}$. Kao primjer, u problemu pretraživanja karte prostora mobilnog robota, pravocrtna udaljenost kao heuristika zadovoljava svojstvo konzistentnosti. Ako je heuristična funkcija dopustiva i ima svojstvo konzistentnosti tada algoritam A* pretražuje najmanji broj čvorova grafa $\mathcal{G}$ nužnih za zajamčeni pronalazak optimalne putanje [48].

Pseudokod algoritma A* dan je algoritmom 2.1. Algoritam A* pretražuje čvorove

od starta. Koristi skup $\mathcal{O}$[2] kojem dodaje one čvorove koje trenutno razmatra. Iz skupa $\mathcal{O}$ oduzima čvor o koji ima minimalnu vrijednost funkcije f (tzv. najbolji čvor). Čvor o proširuje njegovim susjedima tako što u skup $\mathcal{O}$ dodaje one čvorove kojima je cijena f veća od $f(o)$ (inicijalno su cijene svih čvorova beskonačne). Svakom dodanom čvoru n u skup $\mathcal{O}$ odredi vrijednosti cijena $h(n)$ i $f(n)$, prema cijeni njegovog prethodnika $h(o)$ i heuristici $g(n)$. Također postavi pokazivač b prema o jer je o trenutno najbolji čvor prethodnik. Algoritam završava izvođenje kada odabere ciljni čvor G.

Algoritam 2.1: A*(S, G)

```
1:  ∀n ∈ 𝒩, f(n) ← ∞  //Inicijalizacija
2:  𝒪 ← S
3:  h(S) ← 0
4:  g(S) ← heuristika(S, G)
5:  f(S) ← h(S) + g(S)
6:  f_min ← 0
7:  dok 𝒪 ≠ ∅ i f_min ≤ h(G)
8:      f_min ← min{f(n) | n ∈ 𝒪}
9:      o ← n*   //za n* vrijedi f(n*) = f_min
10:     𝒪 ← 𝒪 \ o
11:     za ∀n ∈ 𝒩 tako da {o, n} ∈ ℰ
12:         ako f(n) > f(o)
13:             𝒪 ← 𝒪 ∪ n
14:             h(n) ← h(o) + w_{n,o}
15:             g(n) ← heuristika(n, G)
16:             f(n) ← h(n) + g(n)
17:             b(n) ← o
18:         kraj
19:     kraj
20: kraj
```

Svojstva algoritma A*

Lema 2.1. *Za bilo koji čvor $n \in \mathcal{O}$ i za bilo koju optimalnu putanju $\mathcal{P}^*(S, n)$, postoji čvor $n_k \in \mathcal{O}$ koji je na putanji $\mathcal{P}^*(S, n)$ i za koju vrijedi $h(n_k) = h^*(n_k)$.*

Dokaz. Ako je $S \in \mathcal{O}$ (algoritam A* nije izvršio niti jedan prolaz kroz petlju *dok*), tada je $o = S$ i lema je istinita zbog početnog uvjeta $h(S) = h^*(S) = 0$. Pretpostavimo da $S \notin \mathcal{O}$. Neka je $\mathcal{H}$ skup svih čvorova na putanji $\mathcal{P}^*(S, n)$ koji nisu u skupu $\mathcal{O}$ i za koje vrijedi $h(n_i) = h^*(n_i)$, gdje je $n_i = \mathcal{P}^*[i] \in \mathcal{H}$, $i \leq |\mathcal{P}^*|$. Vrijedi $\mathcal{H} \neq \emptyset$, jer $S \in \mathcal{H}$. Neka je $n_j = \mathcal{P}^*[j]$ element iz $\mathcal{H}$ s najvećim mogućim indeksom putanje j. Vrijedi $n_j \neq n$ budući da je $n \in \mathcal{O}$. Neka je $n_k = \mathcal{P}^*[k]$ sljednik od n_j na putanji $\mathcal{P}^*$, odnosno $k = j+1$. Moguće je da je $n_k = n$. Prema definiciji od h vrijedi $h(n_k) \leq h(n_j) + w_{n_j,n_k}$. Budući da je $n_j \in \mathcal{H}$ vrijedi $h(n_j) = h^*(n_j)$, a budući da je $\mathcal{P}^*$ optimalna putanja vrijedi

[2] U literaturi se koristi pojam engl. *open list*, kao spremište trenutno razmatranih čvorova.

$h^*(n_k) = h^*(n_j) + w_{n_j,n_k}$. Stoga slijedi da je $h(n_k) \leq h^*(n_k)$. Budući da najmanja cijena od S do n_k određena u bilo kojem trenutku nije manja od optimalne cijene $h^*(n_k)$, vrijedi $h(n_k) = h^*(n_k)$. Dodatno, mora vrijediti $n_k \in \mathcal{O}$ prema definiciji skupa $\mathcal{H}$. □

Korolar 2.1. *Neka je $g(n) \leq g^*(n)$ (dopustiva heuristika) za svaki $n \in \mathcal{N}$ i neka algoritam A* nije završio izvođenje. Tada za bilo koju optimalnu putanju $\mathcal{P}^*$ od S do bilo kojeg ciljnog čvora n postoji čvor $o \in \mathcal{O}$ na putanji $\mathcal{P}^*$ s cijenom $f(o) \leq f^*(S)$ ($f^* = h^* + g^*$).*

Dokaz. Prema lemi 2.1, postoji čvor $o \in \mathcal{O}$ na putanji $\mathcal{P}^*$ s cijenom $h(o) = h^*(o)$. Prema definiciji f vrijedi:

$$\begin{aligned} f(o) &= h(o) + g(o) \\ &= h^*(o) + g(o) \\ &\leq h^*(o) + g^*(o). \end{aligned} \tag{2.10}$$

Budući da je $\mathcal{P}^*$ optimalna putanja, za svaki $o \in \mathcal{P}^*$ vrijedi $f^*(o) = f^*(S)$, što upotpunjava dokaz. □

Teorem 2.2. *Ako je $g(n) \leq g^*(n)$ za svaki $n \in \mathcal{N}$ tada je algoritam A* dopustiv.*

Dokaz. Ovaj teorem dokazuje se metodom kontradikcije. Neka A* ne nalazi optimalnu putanju od S do G. Postoje dva slučaja nenalaženja optimalne putanje: 1) algoritam nikada ne završava izvođenje i 2) algoritam završi izvođenje kad dosegne čvor G, koji nema optimalnu cijenu f.

1) Neka je G dohvatljiv od S u konačnom broju koraka i s cijenom $f^*(S)$. Budući da je težina bilo kojeg brida u grafu veća od δ, tada za bilo koji čvor n dalje od K koraka od S, gdje je $K = \dfrac{f^*(S)}{\delta}$, vrijedi $f(n) \geq h(n) \geq h^*(n) > K\delta = f^*(S)$. Prema korolaru 2.1 postoji čvor o na optimalnoj putanji tako da vrijedi $f(o) \leq f^*(S) < f(n)$ i prema algoritmu A* (linije 8-9) čvorovi dalje od K koraka od S neće se proširivati, jer će biti odabran čvor o umjesto n. Prema tome, algoritam A* nije mogao "preskočiti" cilj, nego je mogao zauvijek iznova otvarati iste čvorove unutar K koraka od S. Neka je $\mathcal{D}$ skup svih čvorova dohvatljivih unutar K koraka od S (kojih ima $|\mathcal{D}|$). Algoritam A* dodaje čvorove u $\mathcal{O}$ samo ako im je nova cijena f manja nego kad su prethodno napustili skup $\mathcal{O}$ (linija 12), dakle cijena f će se smanjiti do optimalne cijene putanje koja prolazi kroz čvor. Svaki čvor iz $\mathcal{D}$ može se ponovo ubacivati u skup $\mathcal{O}$ konačan broj puta budući da ima konačan broj putanja od S do n koje prolaze kroz čvorove iz $\mathcal{D}$. Nakon konačnog broja proširivanja čvorova, niti jedan čvor iz $\mathcal{D}$ neće se više dodati u skup $\mathcal{O}$. Prema tome, algoritam završava izvođenje što je u kontradikciji s prvotnom pretpostavkom.

2) Pretpostavimo da algoritam A* završava izvođenje kad dosegne čvor G s cijenom $f(G) = h(G) > f^*(S)$. Prema koloraru 2.1, prije završetka algoritma postojao je čvor $o \in \mathcal{O}$ na optimalnoj putanji s $f(o) \leq f^*(S) < f(G)$. Stoga bi se u tom koraku čvor o odabrao za proširivanje radije nego G, što je u kontradikciji da je A* završio. □

Lema 2.2. *Pretpostavimo da korištena heuristična funkcija ima svojstvo konzistentnosti (2.9). Ako A* u nekom koraku izvođenja oduzme čvor n iz skupa $\mathcal{O}$ tada je $h(n) = h^*(n)$. Kao posljedica, algoritam A* više nikada ne dodaje skupu $\mathcal{O}$ čvor koji je jednom oduzeo iz skupa $\mathcal{O}$.*

Dokaz. Neka se algoritam A* nalazi u jednom koraku izvođenja prije oduzimanja čvora n iz skupa $\mathcal{O}$ ($n \in \mathcal{O}$) i pretpostavimo suprotno, $h(n) > h^*(n)$. Postoji neka optimalna putanja $\mathcal{P}^*(S, n)$. Budući da je $h(n) > h^*(n)$, A* nije pronašao $\mathcal{P}^*$. Prema lemi 2.1 postoji čvor $o \in \mathcal{O}$ na putanji $\mathcal{P}^*$ s $h(o) = h^*(o)$. Ako je $o = n$ lema je dokazana. Inače vrijedi

$$\begin{aligned} h^*(n) &= h^*(o) + g^*(o, n) \\ &= h(o) + g^*(o, n). \end{aligned} \tag{2.11}$$

Dalje prema pretpostavci $h(n) > h^*(n)$ i (2.11) vrijedi

$$h(n) > h(o) + g^*(o, n). \tag{2.12}$$

Dodavanjem $g(n)$ jednadžbi (2.12) slijedi

$$h(n) + g(n) > h(o) + g^*(o, n) + g(n), \tag{2.13}$$

i primijenjujući (2.9) na desnu stranu jednadžbe (2.13) slijedi

$$\begin{aligned} h(n) + g(n) &> h(o) + g(o) \\ f(n) &> f(o), \end{aligned} \tag{2.14}$$

što je u kontradikciji da je A* odabrao čvor n za proširivanje kada je čvor o bio bolji. □

Lema 2.3. *Neka je lista $\mathcal{L} = [n_1, n_2, \ldots, n_{|\mathcal{L}|}]$ slijed čvorova redom koje je oduzimao algoritam A* iz skupa $\mathcal{O}$ i neka vrijedi svojstvo konzistentnosti heuristike (2.9). Ako je $p \leq q$, vrijedi $f(n_p) \leq f(n_q)$, $p, q \in \{1, 2, \ldots, |\mathcal{L}|\}$.*

Dokaz. Neka je n čvor koji je algoritam A* oduzeo iz skupa $\mathcal{O}$ nakon čvora m. Postoje dva slučaja: optimalna putanja od S do n 1) ne prolazi kroz čvor m i 2) prolazi kroz čvor m. Slučaj 1) znači da su čvorovi n i m bili u isto vrijeme u skupu $\mathcal{O}$ u trenutku kad je odabran m kao najbolji čvor (proširivanjem m, skupu $\mathcal{O}$ nije dodan n, nego je u njemu morao biti od prije). Time je lema za slučaj 1) dokazana. Slučaj 2) znači da je algoritam proširivanjem m dodao čvor n skupu $\mathcal{O}$ i odredio $h(n) = h(m) + g^*(m, n)$ ($g^*(m, n) = w_{m,n}$). Prema lemi 2.2 vrijedi $h(n) = h^*(n)$ i $h(m) = h^*(m)$ i dalje,

primjenjujući nejednakost (2.9), slijedi:

$$\begin{aligned} f(n) &= h(n) + g(n) \\ &= h^*(n) + g(n) \\ &= h^*(m) + g^*(m,n) + g(n) \\ &\geq h^*(m) + g(m) \\ &\geq h(m) + g(m) \\ f(n) &\geq f(m). \end{aligned} \tag{2.15}$$

Dokaz je kompletan budući da izraz (2.15) vrijedi za sve parove čvorova n_k i n_{k+1}. □

Korolar 2.3. *Uz pretpostavke prethodne leme, ako je algoritam A* oduzeo čvor n iz skupa $\mathcal{O}$ tada je $f(n) \leq f(S)$.*

Dokaz. Neka je algoritam A* pronašao G. Tada prema lemi 2.3 vrijedi $f(n) \leq f(G) = f^*(G) = f^*(S)$. □

Neka je $\mathfrak{a}^*$ skup svih algoritama sličnih algoritmu A* koji imaju neko pravilo odabira čvora s najmanjom cijenom (može biti više čvorova koji imaju minimalnu cijenu, linije 8-9 algoritma 2.1).

Teorem 2.4. *Neka je A bilo koji dopustivi algoritam koji ne koristi više informacija o prostoru od algoritma u skupu $\mathfrak{a}^*$ i pretpostavimo da vrijedi konzistentnost heuristike korištene u algoritmima iz $\mathfrak{a}^*$. Tada za bilo koji δ graf $\mathcal{G}$ postoji $A^* \in \mathfrak{a}^*$ takav da je svaki čvor proširen algoritmom A^* također proširen algoritmom A i da A^* pretražuje manje čvorova od A.*

Dokaz. Dokaz je dan u [48]. □

Složenost A* algoritma kao nedostatak Broj proširenih čvorova algoritmom A* raste eksponencijalno s duljinom optimalne putanje. Pokazano je da se eksponencijalni porast javlja u slučaju da pogreška heurističke funkcije raste brže od logaritma stvarne cijene putanje. Subeksponencijalni porast je ispunjen ako vrijedi

$$|g(n) - g^*(n)| < O(\log g^*(n)), \tag{2.16}$$

Za gotovo sve heuristike u praktičnoj upotrebi, pogreška je barem proporcionalna cijeni putanje. Međutim, primjena dobre heuristike još uvijek uveliko štedi memoriju u usporedbi s uniformnim pretraživanjem.

Primjena algoritma A* na mrežastim kartama zauzeća

Algoritam A* primjenjuje se na mrežastim kartama zauzeća tako što se koristi prethodno opisani graf $\mathcal{G}(\mathcal{N}, \mathcal{E}, \mathcal{W})$. Postoje brojne heuristike koje možemo koristiti na mrežastoj

karti zauzeća. Najčešće korištena heuristična funkcija je Euklidska norma

$$heuristika(n, g) = \|n - g\|, \tag{2.17}$$

gdje su $n, g \in \mathcal{N}$. Međutim, računska složenost računanja korijena i kvadrata u Euklidskoj normi značajna je u usporedbi s proširivanjem nekolicine dodatnih čvorova. Optimalna procedura koja pretražuje minimalan broj čvorova ne mora biti optimalna po utrošenim resursima. Heuristiku možemo odrediti kombinirajući više dopuštenih heuristika određujući je po maksimalnoj vrijednosti iz skupa dopuštenih heuristika kako bi se sačuvalo svojstvo konzistentnosti. Prihvatljivija je heuristika koja koristi cjelobrojnu aritmetiku. U mrežastoj karti zauzeća cijene prijelaza moguće je opisati cjelobrojnim vrijednostima. Primjerice, duljina stranice polja $e_{polje} = 10$, a duljina dijagonale polja $d_{polje} = 14$. U radu koristimo sljedeću heuristiku:

$$\begin{aligned} a &= \max(\,|x_g - x_n|, |y_g - y_n|\,), \\ b &= \min(\,|x_g - x_n|, |y_g - y_n|\,), \\ heuristika(n, g) &= b \cdot d_{polje} + (a - b) \cdot e_{polje}, \end{aligned} \tag{2.18}$$

gdje su (x_n, y_n) koordinate čvora n, a (x_g, y_g) koordinate čvora g. Ova heuristika ima svojstvo konzistentnosti budući da vrijedi nejednakost trokuta.

A* algoritam je najbolje primjenjiv u pretraživanju *statičkog* prostora mobilnog robota. Statički prostor mobilnog robota je takav prostor u kojemu se ništa ne mijenja u vremenu dok robot prolazi kroz njega. *Dinamički* prostor mobilnog robota je svaki realni prostor u kojem se gibaju prepreke (ljudi, roboti) i mijenja raspored prepreka (pomiču klupe, zatvaraju vrata). Takve promjene prostora upisuju se u mrežastu kartu zauzeća tako što odgovarajuća polja postaju zauzeta, a druga slobodna, a u slučaju korištenja sigurnosne maske cijena mijenjaju se i vrijednosti zauzeća. Promjene se u grafu $\mathcal{G}(\mathcal{N}, \mathcal{E}, \mathcal{W})$ očituju tako što se čvorovi koji predstavljaju zauzeta polja oduzimaju iz skupa $\mathcal{N}$ (jer su čvorovi samo slobodna polja u karti), a polja koja su postala slobodna stvaraju nove čvorove koji se dodaju u skup $\mathcal{N}$. U skladu s izmijenjenim skupom $\mathcal{N}$ predefiniraju se skupovi $\mathcal{E}$ i $\mathcal{W}$. Drugim riječima, nastaje novi graf u kojem je potrebno ponovo izvršiti algoritam A*. Putanja se mora izračunati ponovo kod svake promjene u prostoru, što može biti računski prezahtjevno.

Pseudokod algoritma A* primjenjenog u dinamičkim prostorima mobilnog robota dan je algoritmom 2.2. Robot senzorom otkriva promjene u prostoru i izgrađuje novi graf $\mathcal{G}_R$ (linija 3). Ponovnim pozivanjem cijelog algoritma A* od trenutnog čvora R, u kojem se robot nalazi u trenutku kada je detektirao odstupanje stvarnog prostora od karte, izračunava se nova optimalna putanja do cilja G.

Ovakvo je planiranje putanje optimalno po cijeni putanje, ali je neučinkovito sa stajališta utroška memorije i vremena, naročito za velike karte prostora.

Mnoge strategije pretraživanja primjenjive su samo u statičkom prostoru i uz uvjet da robot ima točnu mrežastu kartu zauzeća koja opisuje prostor. Međutim, pretraživa-

Algoritam 2.2: gibanje-robota-A*(S, G)

```
1:  R ← S
2:  dok R ≠ G
3:      G_R ← senzori(R)
4:      ako G_R ≠ G ili R = S
5:          G ← G_R
6:          A*(R, G)
7:      kraj
8:      R ← n | b(n) = R  //Određivanje sljedećeg čvora na putanji
9:  kraj
```

nje prostora koji se u vremenu mijenjaju zahtijeva brzo preračunavanje novih putanja. Razni pristupi u literaturi pokušavaju ispuniti ovaj zahtjev. Postoje verzije algoritma A* koje pokušavaju uštediti na vremenu planiranja tako što povećavaju cijenu putanje. Putanja do cilja izračunava se brže od algoritma A* tako što povećavaju težinu procjene cijene putanje do cilja $g(n)$ [101]. Takva putanja može biti neoptimalna ako nije ispunjeno svojstvo dopustivosti heuristike. Ovi algoritmi spadaju u kategoriju otežanih algoritama A*, kao što su A*$_\epsilon$ [91] (koji dopušta povećanje cijene putanje za faktor $(1+\epsilon)$), svevremenski A* (engl. *Anytime A**) [47, 46] i ARA* [72].

Neki pristupi temelje se na modificiranju prije izračunate putanje, ako putanja sadržava prepreku. Postoje različiti načini modificiranja putanje tijekom gibanja. Jedan od načina je zaobilaženje prepreke koja se nalazi na putanji, sve dok se ne pronađe točka na prepreci koja je najbliža cilju (engl. *Bug algorithm*) [77].

Algoritmi koji učinkovito određuju optimalne putanje u dinamičkim prostorima, a nastavljaju se na algoritam A*, jesu algoritam D* [126] i Fokusirani algoritam D* [127] ili Lagani D* [64] (implementacijski lakši od Fokusiranog algoritma D*). Ime D im dolazi od riječi dinamički. Ovi algoritmi uspoređeni su u [117] i detaljno su opisani u nastavku.

2.3.2 Algoritam D*

Algoritam D* dobro je poznati dopustivi algoritam pretraživanja grafova sa svojstvom brzog izračunavanja optimalne putanje u dinamičkim prostorima [126]. To se svojstvo naziva *dinamičkim planiranjem* (engl. *dynamic planning, replanning*). U [69] ga opisuju kao dinamičku verziju Dijkstrinog algoritma ili dinamičku verziju algoritma A* bez heurističke funkcije ($heuristika(n, g) = 0$). Algoritam D* nalazi optimalnu putanju i u grafovima u kojima se težine bridova mijenjaju.

Algoritam D* svakom proširenom čvoru n određuje vrijednost $g(n)$ (cijena putanje iz čvora n do cilja G) i vrijednost ključne funkcije za proces dinamičkog planiranja, $k(n)$, koja pamti stare vrijednosti $g(n)$ prije promjena težina u grafu.

D* obrađuje promjene cijena putanje za vrijeme procesa pretraživanja procesirajući čvorove *porasta* i *smanjenja*. Čvorovi su svrstani u čvorove porasta ako je $k < g$. Čvorovi

porasta prostiru informaciju o porastu cijene putanje počevši od promijenjenih čvorova i korigiraju cijene svim čvorovima prethodnicima. Čvorovi porasta aktiviraju susjedne čvorove preko kojih se može smanjiti cijena putanje - čvorove smanjenja. Čvorovi su svrstani u čvorove smanjenja ako je $k = g$. Čvorovi smanjenja prostiru informaciju o smanjenju cijene putanje i korigiraju pokazivače kako bi odredili nove optimalne putanje iz čvorova koji su prethodno bili čvorovi porasta.

Izvođenje algoritma dijeli se na *inicijalno izvođenje* i *dinamičko planiranje*. Inicijalno izvođenje algoritma slično je uniformnom pretraživanju (Dijkstrinom algoritmu), odnosno algoritmu A* bez heuristike. Pseudokod algoritma koji tijekom gibanja robota poziva algoritam D* dan je algoritmom 2.3. Prije inicijalnog izvođenja algoritma D* cijene k i g svim čvorovima postavljene su na ∞, a pokazivači iz svakog čvora pokazuju na same sebe (linija 2). Robot se početno nalazi u startnom čvoru ($R = S$).

Algoritam 2.3: gibanje-robota-D*(S, G)

```
1:  R ← S
2:  ∀n ∈ 𝒩, k(n) ← ∞, g(n) ← ∞, b(n) ← n
3:  𝒪 ← ∅ //Inicijalizacija
4:  dodaj-D*(G, 0)
5:  dok R ≠ G
6:      ℐ_R ←promijenjeni-čvorovi(R)
7:      ako R = S ili ℐ_R \ 𝒪 ≠ ∅ //Inicijalno ili dinamičko planiranje
8:          D*(ℐ_R, R)
9:      kraj
10:     R ← b(R)
11: kraj
```

Inicijalno izvođenje U inicijalnom izvođenju algoritam D* pretražuje čvorove od cilja G. Kao i algoritam A*, koristi skup $\mathcal{O}$ u koji dodaje one čvorove koje trenutno razmatra (algoritam 2.4). Počinje dodavanjem ciljnog čvora (linija 4 algoritma 2.3). Pretpostavka je da se prije prvog izvođenja algoritma D* ne događaju nikakve promjene u prostoru pa je poziv algoritma u liniji 8 zbog ostvarenog uvjeta $R = S$.

Algoritam 2.4: dodaj-D*(n, g_{min})

```
1:  ako n ∈ 𝒪
2:      k(n) ← min{g(n), g_min}
3:  inače
4:      k(n) ← min{k(n), g_min}
5:  kraj
6:  g(n) ← g_min
7:  𝒪 ← 𝒪 ∪ n
```

Pseudokod algoritma D* dan je algoritmom 2.5. Inicijalno izvođenje algoritma slično je algoritmu A* uz heurističnu funkciju jednaku 0 (dopustiva i konzistentna). Iz skupa

Algoritam 2.5: D*$(\mathcal{I}_R, S)$

```
1:  za ∀n ∈ I_R \ O | b(n) ≠ n
2:      dodaj-D*(n, g(n))
3:  kraj
4:  k_min ← 0
5:  dok O ≠ ∅ i k_min ≤ g(S)
6:      k_min ← min{k(n) | n ∈ O}
7:      o ← n*   //za n* vrijedi k(n*) = k_min
8:      O ← O \ o
9:      ako k_min < g(n)
10:         za ∀n ∈ N tako da {o, n} ∈ E
11:             ako g(n) ≤ k_min i g(o) > g(n) + w_o,n
12:                 g(o) ← g(n) + w_o,n
13:                 b(o) ← n
14:             kraj
15:         kraj
16:     kraj
17:     ako k_min = g(n)
18:         za ∀n ∈ N tako da {o, n} ∈ E
19:             ako (b(n) = o i g(n) ≠ g(o) + w_n,o) ili (g(n) > g(o) + w_n,o)
20:                 dodaj-D*(n, (g(o) + w_n,o))
21:                 b(n) ← o
22:             kraj
23:         kraj
24:     inače
25:         za ∀n ∈ N tako da {o, n} ∈ E
26:             ako b(n) = n ili (b(n) = o i g(n) ≠ g(o) + w_n,o)
27:                 dodaj-D*(n, (g(o) + w_n,o))
28:             inače ako (b(n) ≠ o i g(n) > g(o) + w_n,o) i o ∉ O
29:                 dodaj-D*(o, g(o))
30:             inače ako (b(n) ≠ o i g(o) > g(n) + w_o,n) i n ∉ O i k_min < g(n)
31:                 dodaj-D*(n, g(n))
32:             kraj
33:         kraj
34:     kraj
35: kraj
```

$\mathcal{O}$ oduzima čvor o koji ima najmanju vrijednost funkcije k. U inicijalnom je izvođenju algoritma 2.5 uvijek ispunjen uvjet $k_{min} = g(n)$ pa se izvršavaju linije 17-24, budući da se prema linijama 6 i 19 neće dodati u skup $\mathcal{O}$ čvor koji će sebi povećati cijenu g. Čvor o proširuje njegovim susjedima tako što u skup $\mathcal{O}$ dodaje one čvorove kojima se može smanjiti cijena g pomoću čvora o. Svakom dodanom čvoru n u skup $\mathcal{O}$ odredi vrijednosti cijena $g(n)$ i $k(n)$, prema cijeni svog prethodnika $g(o)$. Također postavi pokazivač b prema o jer je o trenutno najbolji čvor prethodnik. Algoritam završava inicijalno izvođenje kada odabere čvor S.

Po završetku inicijalnog pretraživanja svi čvorovi koji su prošireni i više nisu u skupu

$\mathcal{O}$ imaju $h = k$ i ta cijena je optimalna prema lemi 2.2. Budući da pretraživanje počinje od čvora G, proračunane su sve optimalne putanje prema čvoru G iz proširenih čvorova. Tu informaciju koristi proces dinamičkog planiranja.

Čvorovi koje algoritam D* nije proširio imaju pokazivač na same sebe ($b(n) = n$). Ta se informacija koristi samo u dinamičkom planiranju algoritma tako što nejednakost $b(n) \neq n$ označuje da je čvor bio proširen u nekom prijašnjem izvođenju algoritma 2.5 (linije 1 i 26). Ako bi se izvršilo iscrpno pretraživanje (tako da se prošire svi čvorovi iz kojih postoji put do G), tada u liniji 5 treba ostaviti samo uvjet $\mathcal{O} \neq \emptyset$.

Dinamičko planiranje Daljnji izračun algoritma D* ovisi o informaciji senzora o prostoru tijekom gibanja robota. Algoritam se ponovo izvodi samo u slučaju promjene vrijednosti zauzeća nekog polja, koje je robot otkrio senzorom. Funkcija promijenjeni-čvorovi(R) u algoritmu 2.3 vraća sve čvorove kojima se promijenila vrijednost $z(\cdot)$. Ako postoje polja kojima se promijenila vrijednost zauzeća s ∞ na vrijednost $< \infty$, onda se moraju dodati u skup čvorova $\mathcal{N}$. Polja koja su postala zauzeta ($z(\cdot) = \infty$) ostaju u skupu čvorova $\mathcal{N}$ radi ažuriranja cijena njihovim susjedima. Težine bridova koji sadrže te čvorove bit će ∞ prema (2.2).

Čvorovi kojima se promijenila vrijednost z dodaju se skupu $\mathcal{O}$, uz uvjet da je algoritam proširio ove čvorove u prijašnjim izvođenjima ($b(n) \neq n$). Ovo opisuju linije 1-3 algoritma 2.5. Među njima je čvor s najmanjom vrijednosti k, budući da se nalaze usred pretraženog područja pa je njihova cijena manja od čvorova na rubu pretraženog područja (lema 2.3). Njihova cijena je optimalna jer su bili prošireni u prethodnom izvođenju algoritma, a ne nalaze se u skupu $\mathcal{O}$ (lema 2.2). Prema tome, za prvi odabrani najbolji čvor vrijedi $k = g$. On prostire promjenu cijene (porast ili smanjenje) na čvorove prethodnike (čvorovima koji su za njega vezani pokazivačem $b(\cdot)$) ili smanjuje cijenu susjednim čvorovima koji nisu prethodnici (linije 17-24). Ako je promjena u prostoru bila porast vrijednosti zauzeća, nastat će čvorovi porasta. Ti čvorovi dalje šire porast susjednim čvorovima prethodnicima (linije 24-34). Porast se širi prema trenutnoj poziciji robota S. U svakoj se iteraciji algoritma D* provjerava može li se najboljem čvoru porasta smanjiti cijena (linije 9-16). Također se provjerava može li se susjednom čvoru, koji nije prethodnik najboljem čvoru, smanjiti cijena (linije 28-30). Tada se najbolji čvor dodaje u skup $\mathcal{O}$ za ponovno razmatranje. Ovim se korakom izbjegava stvaranje zatvorenih petlji u pokazivačima. Također se provjerava može li susjedni čvor, koji nije prethodnik najboljem čvoru, smanjiti cijenu najboljem čvoru (linije 30-32). Tada se susjedni čvor dodaje u skup $\mathcal{O}$ za kasnije razmatranje. Odgađa se ažuriranje cijene susjednog čvora jer najbolji čvor porasta nema optimalnu cijenu ($k < g$). Kada se pojavi najbolji čvor smanjenja, on svim svojim prethodnicima širi smanjenje cijene, a čvorovima koji nisu prethodnici korigira cijenu (linije 17-24). Čvorovi smanjenja preusmjeravaju pokazivače i stvaraju nove optimalne putanje. Pretraživanje traje dokle god postoji čvor u skupu $\mathcal{O}$ kojem je cijena k manja nego čvoru S. Broj pretraženih čvorova je najmanji mogući, a kao posljedica i vrijeme izračuna.

2.3.3 Fokusirani algoritam D*

Algoritam FD* nastavlja se na algoritam D* tako što koristi heuristiku za procjenu cijene puta od nekog čvora n do S [127]. Algoritam FD* naziva se dinamičkom inačicom algoritma A*. Heuristikom nastoji smanjiti ukupni broj pretraženih čvorova u odnosu na algoritam D*. Algoritam FD*, kao i algoritam A*, pretražuje najmanji broj čvorova nužan za pronalaženje optimalnog puta od starta do cilja ako je heuristična funkcija dopustiva i ima svojstvo konzistentnosti. Međutim, glavna prednost algoritma FD* leži u sposobnosti brzog planiranja putanje u dinamičkim prostorima. Algoritam FD*, kao i D*, nalazi optimalnu putanju u grafovima u kojima se težine bridova mogu mijenjati.

Algoritam 2.6: FD*$(\mathcal{I}_R, S)$

za $\forall n \in \mathcal{I}_R \setminus \mathcal{O} \mid b(n) \neq n$
  dodaj-FD*$(n, g(n))$
kraj
$f_{min} \leftarrow 0$
dok $\mathcal{O} \neq \emptyset$ i $f_{min} \leq g(S)$
  $f_{min} \leftarrow \min\{f(n) \mid n \in \mathcal{O}\}$
  $o \leftarrow n^*$ *//za* n^* *vrijedi* $f(n^*) = f_{min}$
  $\mathcal{O} \leftarrow \mathcal{O} \setminus o$
  ako $k_{min} < g(n)$
    za $\forall n \in \mathcal{N}$ tako da $\{o, n\} \in \mathcal{E}$
      ako $[(h(n) + g(n)), g(n)] \leq [f(o), k(o)]$ i $g(o) > g(n) + w_{o,n}$
        $g(o) \leftarrow g(n) + w_{o,n}$
        $b(o) \leftarrow n$
      kraj
    kraj
  kraj
  ako $k_{min} = g(n)$
    za $\forall n \in \mathcal{N}$ tako da $\{o, n\} \in \mathcal{E}$
      ako $(b(n) = o$ i $g(n) \neq g(o) + w_{n,o})$ ili $(g(n) > g(o) + w_{n,o})$
        dodaj-FD*$(n, (g(o) + w_{n,o}))$
        $b(n) \leftarrow o$
      kraj
    kraj
  inače
    za $\forall n \in \mathcal{N}$ tako da $\{o, n\} \in \mathcal{E}$
      ako $b(n) = n$ ili $(b(n) = o$ i $g(n) \neq g(o) + w_{n,o})$
        dodaj-FD*$(n, (g(o) + w_{n,o}))$
      inače ako $(b(n) \neq o$ i $g(n) > g(o) + w_{n,o})$ i $o \notin \mathcal{O}$
        dodaj-FD*$(o, g(o))$
      inače ako $(b(n) \neq o$ i $g(o) > g(n) + w_{o,n})$ i $n \notin \mathcal{O}$
      i $[f(o), k(o)] < [(h(n) + g(n)), g(n)]$
        dodaj-FD*$(n, g(n))$
      kraj
    kraj
  kraj
kraj

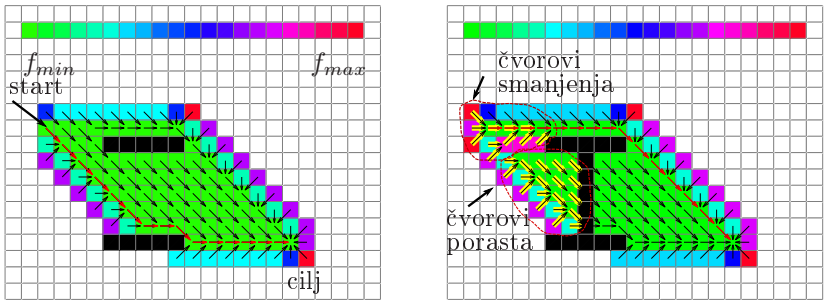
Pseudokod algoritma FD* dan je algoritmom 2.6. Algoritam FD* svakom proširenom čvoru n određuje vrijednost $g(n)$ (cijena putanje iz čvora n do cilja G), vrijednost $k(n)$ (stara vrijednost $g(n)$ prije modifikacije težina u grafu) i vrijednost $f(n) = k(n) + h(n)$, gdje je $h(n)$ procjena cijene optimalne putanje od n do S – heuristična funkcija ($h(n) = heuristika(n, S)$).

Tijekom gibanja robota u algoritmu 2.3 poziva se algoritam FD* umjesto poziva algoritma D*. Kao i algoritam D*, algoritam FD* pretražuje čvorove od cilja G. Koristi skup $\mathcal{O}$ kojem dodaje one čvorove koje trenutno razmatra, kao i algoritmi A* i D*. Algoritam 2.7, za razliku od algoritma D*, računa još i vrijednost funkcije f za svaki dodani čvor (linija 8).

Algoritam 2.7: dodaj-FD*(n, g_{min})

```
1:  ako n ∈ O
2:      k(n) ← min{g(n), g_min}
3:  inače
4:      k(n) ← min{k(n), g_min}
5:  kraj
6:  g(n) ← g_min
7:  h(n) ← heuristika(n, S)
8:  f(n) ← k(n) + h(n)
9:  O ← O ∪ n
```

Daljnji izračun algoritma pokreće nova informacija iz prostora. FD* kao i D* obrađuje promjene cijena u procesu pretraživanja procesuirajući čvorove porasta i čvorove smanjenja. FD* koristi heurističnu funkciju da bi dodatno ubrzao proces dinamičkog planiranja. Iz skupa $\mathcal{O}$ proširuje čvor o koji ima najmanju vrijednost funkcije f (kao algoritam A*). Pretražen je manji broj čvorova, ali je zato i manjem broju čvorova određena nova optimalna putanja. Uključivanje heuristike u pretraživanje postignuto je promjenom linija 4-6, 11 i 30 izvornog algoritma D*. U linijama 11 i 30 uglate zagrade koriste se kao zapis vektora, a usporedba dvaju vektora radi se po komponentama kako slijedi. Za vektore $a = [a_1, a_2]$ i $b = [b_1, b_2]$, vrijedi $a < b$ ako je $a_1 < b_1$ ili $(a_1 = b_1$ i $a_2 < b_2)$.



Slika 2.3. *Primjer pretraživanja prostora algoritmom FD*. Lijevo – inicijalno pretraživanje; Desno – dinamičko planiranje.*

Na slici 2.3 prikazano je inicijalno planiranje (lijevo) i dinamičko planiranje (desno), gdje je optimalna putanja prikazana crvenim strelicama. Rastuće vrijednosti od f prikazane su odgovarajućom bojom počevši od najmanje vrijednosti u cilju do najveće vrijednosti pri rubovima pretraženog područja. Čvorovi porasta propagiraju porast cijene u smjeru trenutne pozicije robota (S). Čvorovi smanjenja preusmjeravaju pokazivače tako da tvore nove optimalne putanje.

2.3.4 Witkowskijev algoritam

Witkowskijev algoritam pripada skupini neinformiranih strategija pretraživanja. Za razliku od općenitog dopustivog algoritma pretraživanja grafa koji oduzima onaj čvor iz skupa $\mathcal{O}$ koji minimizira neku evaluacijsku funkciju (f ili k), ovaj algoritam koristi red FIFO (prvi čvor koji se ubaci u red prvi se i vadi iz reda). Budući da se zasniva na algoritmu BFS (pretraživanje u širinu), pretražuje više čvorova nego algoritam FD* ili algoritam A*. Algoritam je osmišljen za implementaciju na paralelnom računalu na kojemu se svaki čvor paralelno obrađuje pa je vrijeme pretraživanja u takvoj implementaciji proporcionalno duljini putanje. Algoritam BFS nalazi optimalnu putanju samo u grafovima u kojima su sve težine jednakih vrijednosti [107]. BFS uvijek daje putanju s najmanjim brojem koraka (bridova) do cilja. U otežanom grafu takva putanja nije nužno i najjeftinija. Međutim, može se provjeriti da BFS daje optimalnu putanju u grafu stvorenim iz binarne mrežaste karte zauzeća, gdje postoje dvije vrijednosti težina: duljina stranice polja i duljina dijagonale polja. Različite vrijednosti težina ne ruše optimalnost dobivene putanje budući da je duljina dijagonale manja od dviju duljina stranice polja. Stoga će nađena putanja algoritmom BFS koja ima najmanji broj prijelaza usitinu biti i najjeftinija putanja.

Pseudokod Witkowskijeva algoritma dan je algoritmom 2.8. Algoritam pretražuje graf u dva prolaza: *naprijed* i *nazad*. U prolazu naprijed čvorovi se pretražuju od starta prema cilju, dok se u prolazu nazad čvorovi pretražuju od cilja prema startu. Oba su prolaza algoritmi BFS s dodatnim pamćenjem vrijednosti $h(n)$ i $g(n)$ za svaki čvor (minimalna cijena putanje od starta (cilja) do čvora n). Algoritam koristi listu red_S u prolazu naprijed i listu red_G u prolazu natrag. S početka liste oduzima čvor koji proširuje, a na kraj liste dodaje čvorove koje trenutno razmatra (FIFO red). Algoritam završava izvođenje kada se u prolazu naprijed proširi ciljni čvor, a u prolazu nazad startni čvor. Tada cijene putanja g i h za svaki prošireni čvor n daju ukupnu cijenu izračunatu kao zbroj, $f(n) = h(n) + g(n)$ (linija 23). Cijena $f(n)$ je cijena putanje od starta do cilja koja prolazi kroz čvor n i koja je sastavljena od dviju optimalnih putanja: od S do n i od n do G, budući da po završetku oba prolaza algoritma vrijedi $g(n) = g^*(n)$ i $h(n) = h^*(n)$ za svaki prošireni čvor. Najmanja vrijednost f je u čvorovima koji se nalaze na optimalnoj putanji od starta do cilja i iznosi $f_{min} = f(S) = f(G)$.

U slučaju više optimalnih putanja od S do G u mrežastoj karti zauzeća, sve se optimalne putanje sastoje od n ravnih i m dijagonalnih prijelaza, a varijacije u redoslijedu ravnih i dijagonalnih prijelaza čine različite optimalne putanje. Neka je $\mathcal{F}$ skup svih

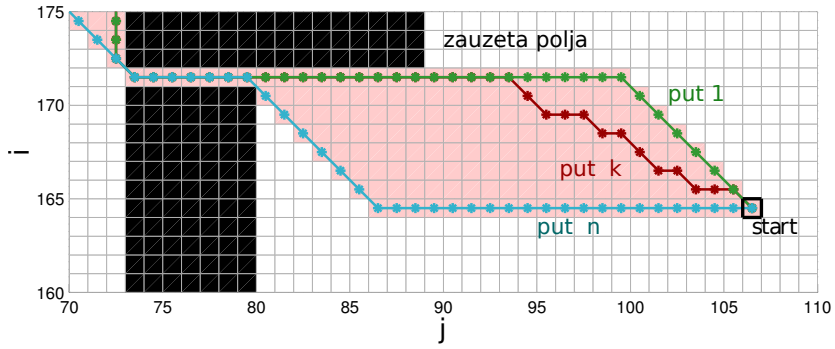
Algoritam 2.8: Witkowski(S, G)

```
1:  ∀n ∈ 𝒩, h(n) ← ∞, g(n) ← ∞, f(n) ← ∞  //Inicijalizacija
2:  red_S ← S  //Prolaz naprijed
3:  h(S) ← 0
4:  dok red_S ≠ ∅ i ∃o ∈ red_S | h(o) ≤ h(G)
5:      o ← red_S[1]
6:      red_S ← red_S \ o
7:      za ∀n ∈ 𝒩 tako da {o, n} ∈ ℰ
8:          ako h(n) > h(o) + w_{n,o}
9:                  red_S ← red_S ∪ n
10:                 h(n) ← h(o) + w_{n,o}
11:         kraj
12:     kraj
13: kraj
14: red_G ← G  //Prolaz nazad
15: g(G) ← 0
16: dok red_G ≠ ∅ i ∃o ∈ red_G | g(o) ≤ g(S)
17:     o ← red_G[1]
18:     red_G ← red_G \ o
19:     za ∀n ∈ 𝒩 tako da {o, n} ∈ ℰ
20:         ako g(n) > g(o) + w_{n,o}
21:                 red_G ← red_G ∪ n
22:                 g(n) ← g(o) + w_{n,o}
23:                 f(n) ← h(n) + g(n)
24:         kraj
25:     kraj
26: kraj
```

čvorova n za koje vrijedi $f(n) = f_{min}$. Da bi se stvorila jedna putanja, zahtijeva se dodatan prolaz kroz čvorove iz skupa $\mathcal{F}$ počevši od starta sve dok se ne dosegne cilj. U slučaju u kojem postoji više od jedne optimalne putanje kroz te čvorove, proizvoljno se odabire jedna od njih, a ostale se napuštaju. Primjerice, može se odabirati uvijek dijagonalan prijelaz kroz čvorove iz skupa $\mathcal{F}$ počevši od S, a ravni prijelaz odabrati ako dijagonalan prijelaz nije mogući. Primjer mnogobrojnih optimalnih putanja generiranih Witkowskijevim algoritmom na mrežastoj karti zauzeća prikazan je na slici 2.4.

Witkowskijev algoritam nema sposobnost brzog dinamičkog planiranja putanje kao algoritmi D* i FD*, nego kao i A* mora ponoviti cijelu proceduru iz početka. Pseudokod Witkowskijeva algoritma koji je primjenjiv u dinamičkim prostorima mobilnog robota dan je algoritmom 2.9. Robot senzorom otkriva promjene u prostoru i izgrađuje novi graf $\mathcal{G}_R$. Ponovnim pozivanjem cijelog Witkowskijeva algoritma od trenutnog čvora R, u kojem se robot nalazi u trenutku kada je detektirao odstupanje stvarnog prostora od karte, izračunavaju se nove vrijednosti f. Prema najmanjoj vrijednosti f koja je u čvorovima na optimalnoj putanji, $f_{min} = f(R) = f(G)$, određuje se skup $\mathcal{F}$. Iz tog se skupa odabire sljedeći čvor na optimalnoj putanji (putanjama) do cilja G.

Glavna je prednost Witkowskijeva algoritma što nalazi sve optimalne putanje koje



Slika 2.4. *Primjer mnogobrojnih optimalnih putanja u mrežastoj karti zauzeća generiranih Witkowskijevim algoritmom. Označene su samo tri optimalne putanje među mnogima.*

Algoritam 2.9: gibanje-robota-W(S, G)

```
1:  R ← S
2:  dok R ≠ G
3:      G_R ← senzori(R)
4:      ako G_R ≠ G ili R = S
5:          G ← G_R
6:          Witkowski(R, G)
7:      kraj
8:      F ← {n | f(n) = f(G)}
9:      R ← odaberi-sljedećeg(R, F)
10: kraj
```

postoje u grafu. U većini slučajeva putanje su jedna do druge, kao na slici 2.4. Jedino u slučaju simetričnih konfiguracija prepreka, putanje mogu biti udaljene jedna od druge, primjerice putanja s lijeva i s desna oko prepreke koja je smještena između starta i cilja. Ako predstavimo sve čvorove iz skupa $\mathcal{F}$ kao skupinu polja koja čine geometrijsko područje koje zovemo područjem skupa $\mathcal{F}$, moguće je sastaviti novu putanju kao niz točaka spojenih ravnim segmentima od kojih svaki leži unutar područja skupa $\mathcal{F}$. Budući da je putanja koja prolazi samim rubom tog područja optimalna za dani graf sastavljen iz mrežaste karte zauzeća, nova putanja koja prolazi sredinom područja kraća je od optimalne putanje u grafu.

Gennery koristi ovu prednost Wiktowskijevog algoritma [41]. Njegov algoritam koristi mrežastu kartu zauzeća s realnim vrijednostima cijena prijelaza (težina) zasnovanim na stereo viziji zemljine plohe. Cijena prelaženja polja sastoji se od kombinacije prevaljene udaljenosti i vjerojatnosti da je polje neprohodno (zauzeto). Putanja koju bi pronašao Witkowskijev algoritam u takvom grafu ne bi bila optimalna. Gennery je proširio Witkowskijev algoritam tako da nalazi optimalne putanje u grafovima s proizvoljnim cijenama prijelaza. U njegovom algoritmu prolazi naprijed i nazad nastavljaju pretraživanje čvorova i nakon što su cilj i start pronađeni. Pretraživanje završava kada cilj (start) ima najmanju vrijednost g (h) među svim pretraženim čvorovima. Ovime

je značajno produljeno vrijeme izračuna. Nakon toga Genneryjev algoritam određuje putanju sastavljenu od ravnih linijskih segmenata od kojih svaki leži unutar područja proširenog skupa $\mathcal{F}'$ koji uključuje sve čvorove kojima je cijena f manja ili jednaka nekoj zadanoj vrijednosti $f'_{min} > f_{min}$. Putanja je određena metodom iterativnog uklapanja krajnjih točaka linijskih segmenata u to područje. Međutim, optimalnost takve putanje nije zajamčena.

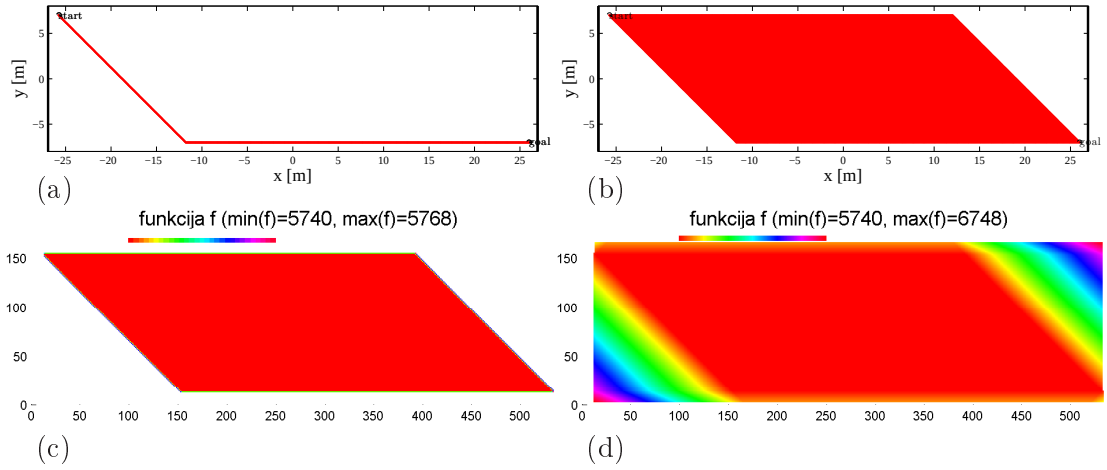
2.3.5 Usporedba algoritma FD* i Witkowskijeva algoritma

Tablica 2.1. *Usporedba algoritma FD* i Witkowskijeva algoritma (W) pretraživanja na trima različitim kartama prostora.*

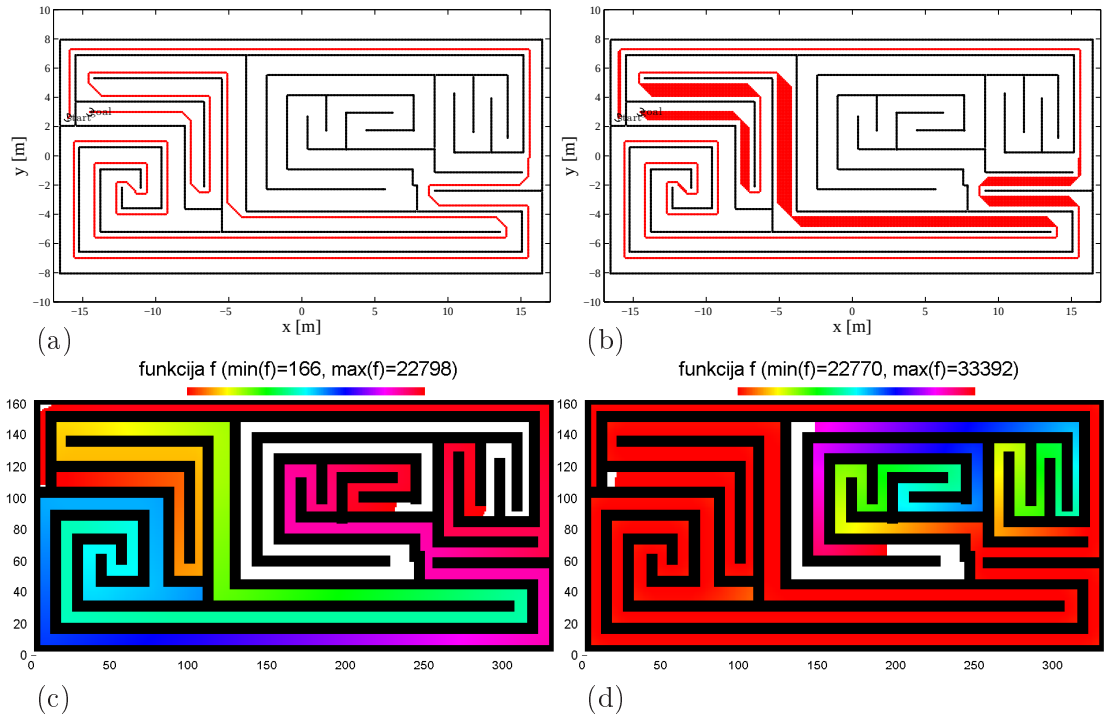
Algoritam	karta	duljina putanje [br. polja]	cijena putanje [cm]	broj pretraženih čvorova	najveći broj čvorova u $\mathcal{O}$ (red_S, red_G)	vrijeme pretraživanja [ms]
FD*	prazna	519	5740	54760	1322	278
W	prazna	519	5740	119495	327, 326	127
FD*	labirint	2246	22770	21259	310	46
W	labirint	2246	22770	70237	44, 43	40
FD*	zavod	248	2690	7456	206	25
W	zavod	248	2690	47471	197, 103	19

U Tablici 2.1 dani su rezultati pretraživanja algoritma FD* i Witkowskijeva algoritma (W) na trima različitim kartama prostora: karta bez prepreka (prazna), labirint i karta Zavoda za automatiku i računalno inženjerstvo FER-a. Oba algoritma daju za svaku kartu međusobno jednake duljine i cijene putanja. Cijena optimalne putanje dobivena algoritmom FD* je vrijednost funkcije f u startnom čvoru (startni čvor je završni čvor pretraživanja u kojem je vrijednost heuristike 0). Cijena optimalne putanje dobivena Witkowskijevim algoritmom je vrijednost funkcije f u svim čvorovima koji su na optimalnoj putanji. U slučaju da postoji više optimalnih putanja, Witkowskijev algoritam nalazi sve takve putanje. Algoritmi se razlikuju po broju pretraženih čvorova. Witkowskijev algoritam pretražuje više čvorova od algoritma FD* budući da pretražuje u dva smjera i proširuje sve čvorove, a ne samo najbolje prema nekoj kriterijskoj funkciji. Iako proširuje više čvorova, vrijeme pretraživanja je manje. Razlog leži u tome što kriterij prema kojem FD* odabire redoslijed proširivanja čvorova zahtijeva dodatni utrošak vremena (sortiranje ili traženje najboljeg čvora) koji ovisi o broju čvorova koje trenutno ima na raspolaganju (u skupu $\mathcal{O}$). Vrijeme pretraživanja za oba algoritma manje je u prostorima s više prepreka budući da se pretražuju samo slobodna polja mrežaste karte zauzeća.

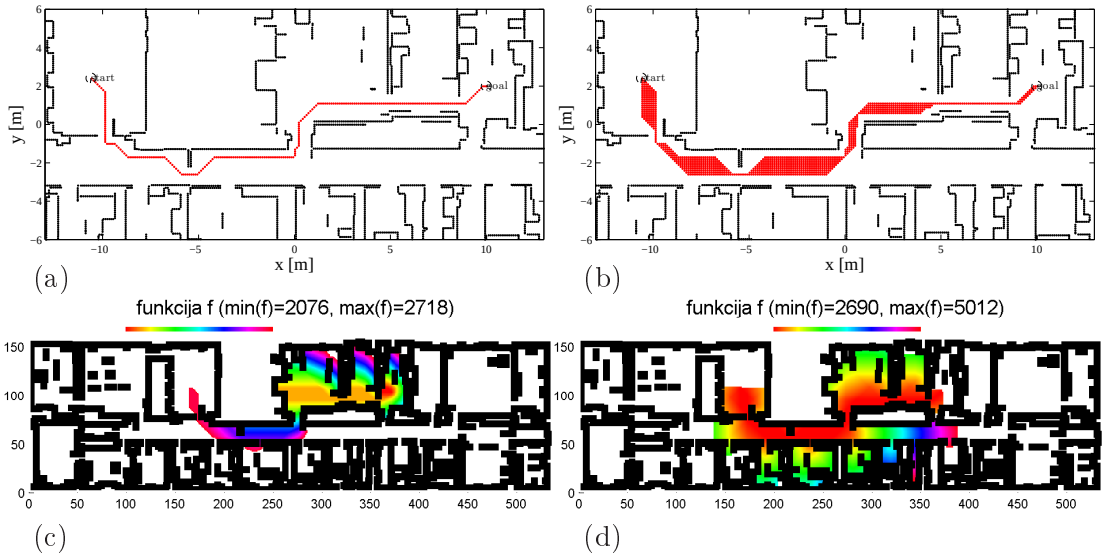
Na slikama 2.5, 2.6 i 2.7 dane su izračunate optimalne putanje za svaku kartu prostora i dani su prikazi izračunatih vrijednosti funkcije f za svaki čvor u pretraženom području. Na slikama su naznačeni startni i ciljni čvorovi. Svaka je vrijednost funkcije



Slika 2.5. *Karta bez prepreka: izračunate putanje (a) algoritmom FD* i (b) Witkowskijevim algoritmom; i izračunate vrijednosti f (c) algoritmom FD* i (d) Witkowskijevim algoritmom.*



Slika 2.6. *Labirint: izračunate putanje (a) algoritmom FD* i (b) Witkowskijevim algoritmom; i izračunate vrijednosti f (c) algoritmom FD* i (d) Witkowskijevim algoritmom.*



Slika 2.7. *Karta Zavoda: izračunate putanje (a) algoritmom FD* i (b) Witkowskijevim algoritmom; i izračunate vrijednosti f (c) algoritmom FD* i (d) Witkowskijevim algoritmom.*

f prikazana drugom bojom i prikazana je legenda boja u obliku trake s kontinuiranom promjenom boje s lijeva na desno - od najmanje do najveće vrijednosti funkcije f. Vrijednosti f u Witkowskijevu algoritmu predstavljaju točan iznos cijene najkraće putanje od starta do cilja kroz neki čvor. Vrijednosti f u algoritmu FD* su procjena cijene putanje od starta do cilja kroz neki čvor zbog heuristike koja se u proračunu koristi. U slučaju karte bez prepreka (slika 2.5) vrijednosti funkcije f se poklapaju za oba algoritma za veći dio prostora. To je dio prostora u kojem postoji više optimalnih putanja. Budući da se radi o karti bez prepreka, korištena je heuristika (2.18) u algoritmu FD* točno procijenila cijenu preostalog dijela putanje i stoga je vrijednost f točna. U slučajevima karata s preprekama vidljivo je kako vrijednost f algoritma FD* raste od ciljnog čvora do startnog čvora, što je ostvareno zbog konzistentnosti heurističke funkcije (lema 2.3). Naročito je vidljiv znatan raspon vrijednosti f u slučaju labirinta (slika 2.6c), gdje su start i cilj postavljeni blizu po međusobnoj udaljenosti, a zapravo daleko po putanji u slobodnom prostoru pa heuristika loše procjenjuje cijenu putanje.

2.4 Dvosmjerni algoritam D* dinamičkog planiranja putanje

Algoritmi planiranja putanje zasnovani na mrežastim kartama imaju nedostatak da je dobivena putanja sastavljena od niza ravnih segmenata kojima su orijentacije višekratnici od 45°. Izravno zadavanje takve krivulje kao referentne trajektorije mobilnom robotu (tj. geometrijske putanje parametrirane u vremenu), nije zadovoljavajuće zbog

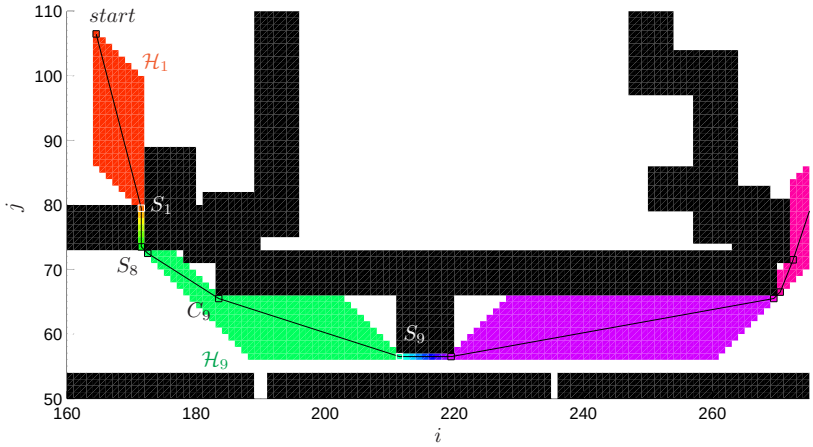
kinematičkih i dinamičkih ograničenja mobilnoga robota. Naime, praćenje ravnih segmenata kojima se smjer naglo mijenja mora rezultirati znatnim usporavanjem gibanja mobilnoga robota. U literaturi postoje brojne metode koje rješavaju taj nedostatak. Primjerice, ekstenzija algoritma D* tzv. algoritam Field D* [31] koristi linearnu interpolaciju za izvođenje cijena putanje svih točaka u prostoru, a ne samo onih na sjecištima mreže. Tako učinkovito nalazi prirodnije i kraće putanje s kontinuiranim vrijednostima orijentacija odsječaka putanje. Zanimljivost algoritma Field D* jest da povećanjem razlučivosti mrežaste karte putanja postaje slična Euklidskoj najkraćoj putanji, međutim optimalnost putanje nije zajamčena. Ovaj algoritam ima otprilike dvaput dulje vrijeme izračuna u odnosu na algoritam D*. Postoje i drugi algoritmi koji interpolacijom određuju cijene putanje točaka izvan diskretnog skupa uzoraka, primjerice dinamičko programiranje s interpolacijom [69] ili već spomenuti Genneryjev algoritam [41].

Inspirirani Wiktowskijevim algoritmom, izveli smo algoritam planiranja putanje, nazvan dvosmjernim algoritmom D* (DD*), koji koristi dva prolaza algoritma D* kroz graf pretraživanja - prvi prolaz od cilja, a drugi od starta. Za razliku od izvornog algoritma D*, algoritam DD* određuje prirodnije i jeftinije putanje kroz mrežu s rasponom kontinuiranih orijentacija linijskih segmenata putanje. Proizvedena nova putanja, nazvana putanjom $\mathcal{P}_W$, najkraća je moguća putanja u geometrijskom prostoru kada veličina polja teži k nuli.

Ostatak je odlomka organiziran kako slijedi. Odlomak 2.4.1 opisuje postupak određivanja putanje $\mathcal{P}_W$ u binarnoj mrežastoj karti zauzeća za koju je pokazano da je takva putanja najkraća moguća kad veličina polja teži k nuli. Taj je postupak opisan u radu [116]. Nakon toga, u odlomku 2.4.2 prikazana je osnovna ideja algoritma DD* – određivanje putanje $\mathcal{P}_W$ u mrežastoj karti zauzeća s proizvoljnim vrijednostima zauzeća, kakva je mrežasta karta zauzeća sa sigurnosnom maskom cijena. Odlomak 2.4.3 opisuje dinamičko planiranje algoritma DD* u slučaju promjene u prostoru. U odlomku 2.4.4 dane su eksperimentalne provjere algoritma DD* u usporedbi s provjerama algoritma D* pod istim uvjetima.

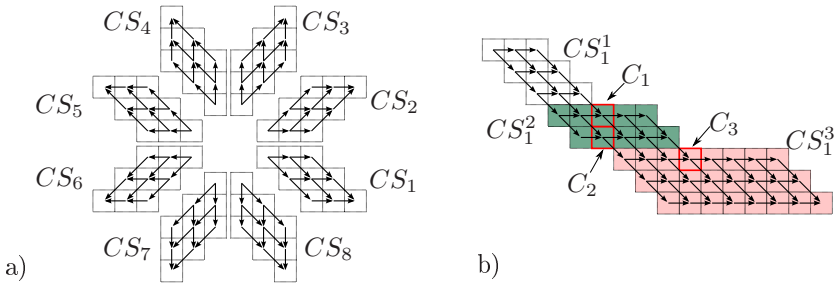
2.4.1 Određivanje najkraće putanje u binarnoj mrežastoj karti zauzeća

Pretpostavimo da je Witkowskijevim algoritmom za svaki čvor n u grafu $\mathcal{G}$ stvorenom iz mrežaste karte zauzeća izračunata vrijednost f – cijena putanje od starta do cilja koja prolazi čvorom n. Najmanja je vrijednost f u čvorovima koji se nalaze na optimalnoj putanji od starta do cilja i iznosi $f_{min} = f(S) = f(G)$. Ti čvorovi čine skup $\mathcal{F} = \{n \mid f(n) = f_{min}\}$. Čvorovi skupa $\mathcal{F}$ čine geometrijsko područje koje u nastavku zovemo područjem skupa $\mathcal{F}$. Cilj je napraviti particiju skupa $\mathcal{F}$ na manje skupove, označenih $\mathcal{H}_i$, $i = 1, \ldots, H$ i unutar područja svakog skupa $\mathcal{H}_i$ pronaći linijske segmente putanje takve da čine najkraću moguću putanju $\mathcal{P}_W$ u geometrijskom prostoru, kao na slici 2.8. Svaki podskup $\mathcal{H}_i$, $i = 1, \ldots, H$, sadrži optimalne putanje u grafu $\mathcal{G}$ koje se sastoje od točno dva smjera prijelaza kroz mrežu – jedan ravni smjer i jedan dijagonalni. Svaki



Slika 2.8. *Skupovi čvorova $\mathcal{H}_i$, konkavne točke tipa C i S i linijski segmenti koji ih spajaju određeni na binarnoj mrežastoj karti zauzeća.*

podskup $\mathcal{H}_i$ sadrži tzv. konkavne točke koje su uvijek u središtima polja mrežaste karte zauzeća. Konkavne točke pri kojima je područje skupa $\mathcal{F}$ široko jedno polje označene su sa S, a druge su označene sa C. Kod C točaka moguće je podijeliti podskup $\mathcal{H}_i$ na konveksne podskupove, označene s CS_k, gdje je $k \in \{1, \ldots, 8\}$, odnosno postoji točno osam tipova konveksih skupova ovisno o optimalnim putanjama koje prolaze kroz te skupove. Svih osam mogućih konveksnih skupova CS_k prikazano je na slici 2.9.a, a podijela skupa $\mathcal{H}_i$ na konveksne podskupove prikazana je na slici 2.9.b. Svaki konveksni skup CS_k definiraju dva optimalna prijelaza kroz mrežu. Linijski segmenti putanje



Slika 2.9. *a) Osam mogućih tipova konveksnih skupova CS_k, $k = 1, \ldots, 8$; b) Primjer podijele skupa $\mathcal{H}_i$ na konveksne skupove CS_1^j, $j = 1, 2, 3$, s naznačenim konkavnim točkama C i S tipa.*

$\mathcal{P}_W$ spajaju isključivo konkavne točke i leže unutar područja koja pokrivaju polja iz tih konveksnih podskupova. Procedura određivanja najkraće putanje sastoji se od dva koraka: (1) određivanje podskupova čvorova $\mathcal{H}_i$ i konkavnih točaka tipa C i S i (2) određivanje najkraće putanje $\mathcal{P}_W$.

(1) Određivanje podskupova optimalnih putanja i konkavnih točaka

Postupak stvaranja particije skupa $\mathcal{F}$ na skupove $\mathcal{H}_i$, $i = 1, \ldots, H$ i određivanje konkavnih točaka dan je algoritmom 2.10. Koristi se pomoćni skup $\mathcal{H}$ koji je početno jednak

Algoritam 2.10: određivanje-podskupova-optimalnih-putanja($start, cilj, \mathcal{F}$)

```
1:  H ← F
2:  i ← 1
3:  H_i ← start
4:  S_0 ← start  //Početna S točka
5:  C_{H_i} ← ∅  //Skup C točaka pridruženih skupu H_i
6:  H ← H \ start
7:  dok H ≠ ∅ ili cilj ∈ H
8:      O_i ← {n | n ∈ H, m ∈ H_i i {m,n} ∈ E} i g(n) ≤ g(m)}
9:      H_i ← H_i ∪ O_i
10:     H ← H \ O_i
11:     za ∀n ∈ O_i
12:         O_n ← {m | m ∈ F i {m,n} ∈ E}
13:         ako ∄m ∈ O_n tako da |g(n) − g(m)| < w_{n,m}
14:             w_max ← max{w_{m,n} | m ∈ O_n}
15:             ako ∄o ∈ F tako da |g(n) − g(o)| < w_max
16:                 S_i ← n  //Nova S točka
17:                 i ← i + 1
18:                 H_i ← n
19:                 C_{H_i} ← ∅
20:             inače
21:                 C_{H_i} ← C_{H_i} ∪ n  //Nova C točka
22:             kraj
23:         inače ako |O_n| ∈ {5,6} i ∃o ∉ F tako da ‖o − n‖ = e_polje
24:             O_o ← {m | m ∈ F i ‖m − o‖_∞ = e_polje}
25:             ako |O_o| ≤ 4
26:                 C_{H_i} ← C_{H_i} ∪ n  //Nova C točka
27:             kraj
28:         kraj
29:     kraj
30: kraj
```

skupu $\mathcal{F}$. Iz skupa $\mathcal{H}$ se oduzimaju čvorovi i iterativno raspoređuju u podskupove $\mathcal{H}_i$ redom za $i = 1, \ldots, H$, gdje je H konačni broj dobivenih podskupova skupa $\mathcal{F}$. Početni skup $\mathcal{H}_1$ sadrži startni čvor i dalje se u njega iterativno dodaje skup čvorova koji su susjedni čvorovima koji su trenutno u skupu $\mathcal{H}_1$ i čija je cijena g manja ili jednaka cijenama čvorova u skupu $\mathcal{H}_1$ (linija 8). Ovaj je uvjet padajuće vrijednosti cijene g kroz skup $\mathcal{H}_i$ nužan samo u procesu dinamičkog planiranja opisanog u potpoglavlju 2.4.3, u kojem se startni čvor zamijenjuje čvorom R i može nalaziti usred područja skupa $\mathcal{F}$. Popunjavanje skupa $\mathcal{H}_i$ završava kada se doda čvor koji predstavlja konkavnu točku S tipa, označenu sa S_i i tada se započinje popunjavati novi skup $\mathcal{H}_{i+1}$ (linije 16-18). Prva S točka označena s S_0 je *start*, a zadnja S točka je *cilj* i bit će označena s S_H. Da

je neki čvor n točka S provjerava se promatranjem njegovih susjednih čvorova (linija 12). Kroz točku S moraju proći sve optimalne putanje pa tako svi susjedni čvorovi iz skupa $\mathcal{F}$ imaju vrijednost cijene g koja se razlikuje od $g(S)$ točno za težinu w između susjednog čvora i S točke. Prema tome, ako ne postoji susjedni čvor m takav da vrijedi $|g(n) - g(m)| < w_{m,n}$ tada je čvor n kandidat za S točku. Dodatno se mora provjeriti postoji li čvor slične cijene g, ali koji nije susjed kandidatu za S točku. Takav je slučaj u simetričnim konfiguracijama prepreka u kojima postoje dva ili više međusobno udaljenih optimalnih putanja (kao na slici 2.11). Za test sličnosti koristi se najveća težina w_{max} između kandidata za S točku i njegovog susjednog čvora (linija 14). Ako ne postoji čvor $o \in \mathcal{F}$ za koji vrijedi $|g(n) - g(m)| < w_{max}$ tada je čvor n pohranjen kao točka S_i, inače je pohranjen kao C točka budući da ne prolaze sve optimalne putanje kroz taj čvor. Unutar skupa $\mathcal{H}_i$ može postojati jedna ili više konkavnih točaka C tipa. Da je neki čvor n točka C provjerava se promatranjem okoline najbližeg polja o koje polje čvora n dodiruje stranicom (linija 24). Polje o je zauzeto polje u slučaju binarne mrežaste karte zauzeća, inače je to polje kojem je vrijednost zauzeća $z(o) > 1$. Ako polje o ima manje ili jednako četiri okolnja čvora koja pripadaju skupu $\mathcal{F}$ tada se čvor n pamti kao C točka. Sve C točke unutar skupa $\mathcal{H}_i$ dodaju se u skup konkavnih točaka C tipa označen s $C_{\mathcal{H}_i}$ (linija 26). Algoritam završava praznim skupom $\mathcal{H}$. Dodatni uvjet zaustavljanja kad *cilj* više nije u skupu $\mathcal{H}$ postavljen je ako se želi prekinuti stvaranje skupova ranije, u nekoj drugoj točki unutar skupa $\mathcal{F}$, kao što je potrebno u procesu dinamičkog planiranja algoritma DD* opisanog u 2.4.3.

(2) Određivanje najkraće putanje

Linijski segmenti putanje $\mathcal{P}_W$ dobivaju se spajanjem susjednih S točaka, odnosno spajanjem točaka S_{i-1} i S_i, za $i = 1, \ldots, H$, ako između njih ne postoji niti jedna C točka, odnosno ako je skup $\mathcal{H}_i$ konveksan. Inače, stvaranje putanje $\mathcal{P}_W$ uključuje C točke iz skupa $C_{\mathcal{H}_i}$ tako što se određuje najkraća Euklidska putanja od točke S_{i-1} do S_i koja leži unutar područja koje pokrivaju polja iz skupa $\mathcal{H}_i$. U tu se svrhu stvara mali graf pretraživanja za svaki skup $\mathcal{H}_i$, $i = 1, \ldots, H$, označen sa $\mathcal{G}_{\mathcal{H}_i}(\mathcal{N}_{\mathcal{H}_i}, \mathcal{E}_{\mathcal{H}_i}, \mathcal{W}_{\mathcal{H}_i})$. Skup čvorova definiran je kao $\mathcal{N}_{\mathcal{H}_i} = \{S_{i-1}, S_i, C_{\mathcal{H}_i}\}$, odnosno čine ga početna i krajnja S točka skupa $\mathcal{H}_i$ i C točke između tih S točaka koje su prisutne u skupu $C_{\mathcal{H}_i}$. Bridovi su definirani samo između vidljivih čvorova[3], $\mathcal{E}_{\mathcal{H}_i} = \{e_{m,n} := \{m, n\} \mid m, n \in \mathcal{N}_{\mathcal{H}_i},\ m \text{ i } n \text{ su vidljivi}\}$. Svakom bridu $\{m, n\} \in \mathcal{E}_{\mathcal{H}_i}$ pridružena je težina $w_{m,n} := \|m - n\|$. Traži se putanja $\mathcal{P}_i = \mathcal{P}_i(S_{i-1}, S_i)$ u grafu $\mathcal{G}_{\mathcal{H}_i}$ koja će imati najmanju cijenu $c(\mathcal{P}_i)$ među svim mogućim

[3]Dva su čvora vidljiva ako dužina koja ih spaja ne sječe niti jednu prepreku. Uvdje se relacija vidljivosti koristi uz dodatan uvjet da dužina koja ih spaja leži potpuno unutar područja skupa $\mathcal{H}_i$, što je ekvivalentno prvom uvjetu u slučaju binarne mrežaste karte zauzeća.

putanjama u tom grafu. Pri tom je putanja $\mathcal{P}_i$ lista za koju vrijedi:

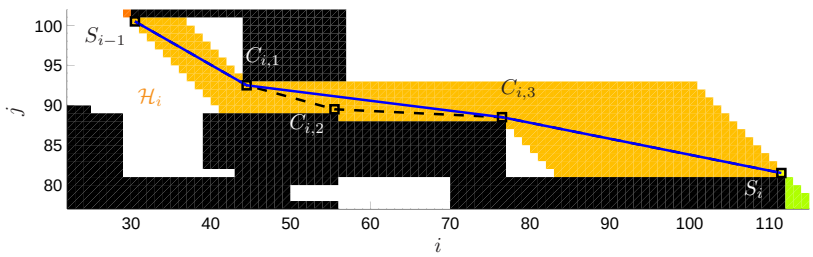
$$\begin{aligned} \mathcal{P}_i[1] = S_{i-1}, \quad \mathcal{P}_i[|\mathcal{P}_i|] = S_i, \\ \mathcal{P}_i[k] \in \mathcal{N}_{\mathcal{H}_i}, \quad k = 1, \ldots, |\mathcal{P}_i|, \\ \{\mathcal{P}_i[l], \mathcal{P}_i[l+1]\} \in \mathcal{E}_{\mathcal{H}_i}, \quad l = 1, \ldots, |\mathcal{P}_i| - 1. \end{aligned} \tag{2.19}$$

Svake dvije susjedne točke putanje $\mathcal{P}_i$ predstavljaju linijski segment koji leži unutar područja skupa $\mathcal{F}$. Cijena putanje $\mathcal{P}_i$ definirana je kao zbroj težina bridova na putanji, odnosno

$$c(\mathcal{P}_i) := \sum_{k=1}^{|\mathcal{P}_i|-1} w_{\mathcal{P}_i[k],\mathcal{P}_i[k+1]}. \tag{2.20}$$

U većini slučajeva graf $\mathcal{G}_{\mathcal{H}_i}(\mathcal{N}_{\mathcal{H}_i}, \mathcal{E}_{\mathcal{H}_i}, \mathcal{W}_{\mathcal{H}_i})$ koji je potrebno pretražiti od S_{i-1} do S_i ima mali broj čvorova i može se koristiti primjerice algoritam A* pretraživanja grafa. Možemo kreirati najgori slučaj u kojem postoji samo jedan skup $\mathcal{H}_1 = \mathcal{F}$, a jedine dvije S točke, S_0 i S_1, su start i cilj. Tada bi za određivanje putanje $\mathcal{P}_W$ trebalo pretražiti samo graf $\mathcal{G}_{\mathcal{H}_1}$, a njegova veličina ovisila bi o broju C točaka. Međutim, broj čvorova u tome grafu još je uvijek znatno manji od broja čvorova u grafu stvorenom iz mrežaste karte zauzeća.

Primjer određivanja putanje $\mathcal{P}_i$ u grafu $\mathcal{G}_{\mathcal{H}_i}(\mathcal{N}_{\mathcal{H}_i}, \mathcal{E}_{\mathcal{H}_i}, \mathcal{W}_{\mathcal{H}_i})$ prikazan je na slici 2.10. Crtkanom linijom označeni su svi bridovi iz skupa $\mathcal{E}_{\mathcal{H}_i}$ koji nisu na najkraćoj putanji,

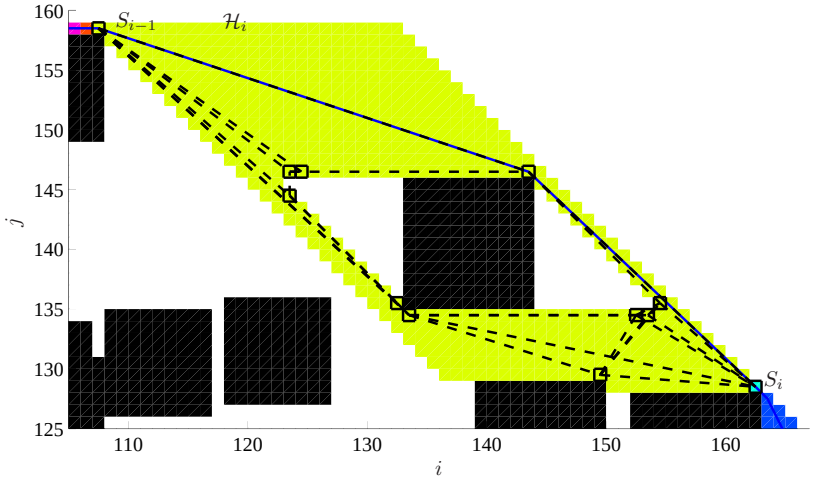


Slika 2.10. *Određivanje putanje $\mathcal{P}_i$ u grafu $\mathcal{G}_{\mathcal{H}_i}(\mathcal{N}_{\mathcal{H}_i}, \mathcal{E}_{\mathcal{H}_i}, \mathcal{W}_{\mathcal{H}_i})$ unutar skupa $\mathcal{H}_i$.*

a punom linijom oni koji su na najkraćoj putanji. U prikazanom primjeru u skupu $\mathcal{H}_i$ postoje tri konkavne točke C tipa označene redom $C_{i,1}$, $C_{i,2}$ i $C_{i,3}$.

Sljedeći primjer na slici 2.11 prikazuje određivanje putanje $\mathcal{P}_i$ u simetričnoj konfiguraciji prepreka u kojoj su optimalne putanje odvojene i okružuju prepreku s obje strane. Drugim riječima, područje skupa $\mathcal{H}_i$ sadrži rupe. Općenito, skup $\mathcal{H}_i$ može sadržavati mnogo rupa, a najkraća putanja se računa istom procedurom kao u slučaju područja bez rupa. U prikazanom primjeru postoji 10 konkavnih točaka C tipa označenih kvadratima. Skup $\mathcal{E}_{\mathcal{H}_i}$ sadrži 29 vidljivih bridova (niti jedan ne prolazi kroz rupu) označenih crtkanom linijom. Najkraća putanja (označena punom linijom) određena je pretraživanjem grafa $\mathcal{G}_{\mathcal{H}_i}(\mathcal{N}_{\mathcal{H}_i}, \mathcal{E}_{\mathcal{H}_i}, \mathcal{W}_{\mathcal{H}_i})$.

Ukupna putanja od starta do cilja određena je unijom svih putanja dobivenih unutar



Slika 2.11. *Određivanje putanje $\mathcal{P}_i$ u grafu $\mathcal{G}_{\mathcal{H}_i}(\mathcal{N}_{\mathcal{H}_i}, \mathcal{E}_{\mathcal{H}_i}, \mathcal{W}_{\mathcal{H}_i})$ unutar skupa $\mathcal{H}_i$.*

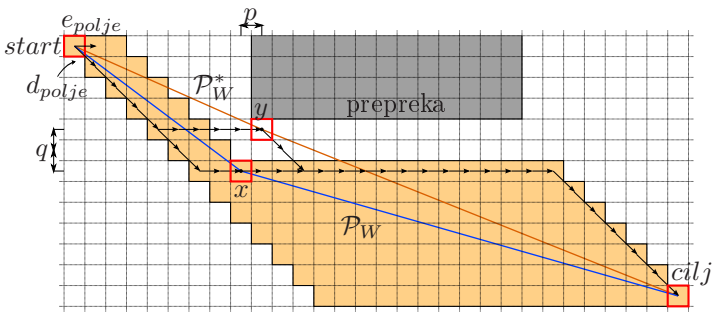
skupova $\mathcal{H}_i$ za $i = 1, \ldots, H$

$$\mathcal{P}_W = \mathcal{P}_1 \cup \cdots \cup \mathcal{P}_H. \tag{2.21}$$

Teorem 2.5. *Putanja $\mathcal{P}_W$ određena u binarnoj mrežastoj karti zauzeća je najkraća putanja u geometrijskom prostoru uz uvjet da su lomne točke putanje u središtima polja mrežaste karte zauzeća. Dodatno, putanja $\mathcal{P}_W$ konvergira najkraćoj Euklidskoj putanji, ako veličina polja teži k nuli.*

Dokaz. Prema opisanom postupku putanja $\mathcal{P}_W$ leži unutar područja skupa $\mathcal{F}$, a kojoj su lomne točke u konkavnim točkama, gdje su sve konkavne točke S i C tipa u središtima polja. Takva je putanja kraća od putanje određene algoritmom D* budući da linijski segmenti prolaze kroz sredinu područja skupa $\mathcal{F}$, a putanja određena algoritmom D* prolazi rubom tog područja. Očito je da je putanja $\mathcal{P}_W$ najkraća među svim mogućim putanjama koje prolaze isključivo područjem skupa $\mathcal{F}$, uz uvjet da se lomne točke nalaze u središtima polja. Međutim, nije očito da ne postoji kraća putanja, označena s $\mathcal{P}_W^*$, uz uvjet da su lomne točke u središtima polja, čiji bi (a) čvorovi bili izvan područja skupa $\mathcal{F}$ ili (b) linijski segmenti prolazili izvan područja skupa $\mathcal{F}$.

Razmotrimo prvo slučaj (a) prikazan na slici 2.12. Neka postoji samo jedan skup $\mathcal{H}_1 = \mathcal{F}$ između startnog i ciljnog čvora. Pretpostavimo da postoji kraća putanja $\mathcal{P}_W^*$ čija se točka y nalazi izvan skupa $\mathcal{H}_1$. Točka y se mora nalaziti s konkavne strane područja i blizu vrha prepreke da bi putanja $\mathcal{P}_W^*$ bila kraća od putanje $\mathcal{P}_W$. Točka x je konkavna točka skupa $\mathcal{H}_1$ i nalazi se na putanji $\mathcal{P}_W$. Tada vrijedi $f(y) > f(x)$, budući da je y izvan skupa $\mathcal{H}_1$. Nadalje, $f(x) = h(x) + g(x)$, gdje je g cijena putanje od cilja do čvora x, a h cijena putanje od starta do čvora x u grafu $\mathcal{G}$ stvorenom iz mrežaste karte zauzeća. Cijena putanje u grafu $\mathcal{G}$ sastoji se od linearne kombinacije ravnih i dijagonalnih prijelaza. Na slici 2.12 naznačena su dva smjera prijelaza optimalnih putanja u grafu $\mathcal{G}$ od kojih se sastoji skup $\mathcal{H}_1$, jedan ravni (e_{polje}) i jedan dijagonalni



Slika 2.12. *Provjera može li čvor na najkraćoj putanji biti izvan područja skupa $\mathcal{F}$.*

(d_{polje}) prijelaz. Neka je optimalna putanja od starta do čvora x sastavljena od n_S ravnih i m_S dijagonalnih prijelaza i neka je optimalna putanja od cilja do čvora x sastavljena od n_G ravnih i m_G dijagonalnih prijelaza. Tada je

$$\begin{aligned} f(x) &= h(x) + g(x) \\ &= n_S \cdot e_{polje} + m_S \cdot d_{polje} + n_G \cdot e_{polje} + m_G \cdot d_{polje}, \end{aligned}$$

$n_S, m_S, n_G, m_G \in \mathbb{N}$. Neka je y smješten p polja desno od x i q polja iznad x. Tada se $f(y)$ računa kao:

$$\begin{aligned} f(y) &= h(y) + g(y) \\ &= (n_S + q + p) \cdot e_{polje} + (m_S - q) \cdot d_{polje} \\ &\quad + (n_G - q - p) \cdot e_{polje} + (m_G + q) \cdot d_{polje}, \end{aligned} \tag{2.22}$$

$p, q \in \mathbb{N}$. Sada je jasno da je $f(y) = f(x)$, što je u kontradikciji s pretpostavkom da je y izvan skupa $\mathcal{H}_1$. Ovime je pokazano da su konkavne točke područja skupa $\mathcal{F}$ uvijek kod prepreka i udaljene od prepreka za pola stranice polja. Stoga, u slučaju (b), ako bi linijski segmenti putanje $\mathcal{P}_W^*$ prolazili izvan tog područja morali bi prolaziti s konveksne strane područja jer bi inače prolazili kroz prepreku. Međutim, prolazak putanje $\mathcal{P}_W^*$ s konveksne strane područja daje dulju putanju nego prolaskom kroz samo područje što je u kontradikciji s pretpostavkom da je putanja $\mathcal{P}_W^*$ kraća od $\mathcal{P}_W$. Budući da je pokazano da najkraća putanja mora prolaziti kroz konkavne točke, primjer na slici 2.12 može se poopćiti na proizvoljan položaj starta i cilja i broj podskupova $\mathcal{H}_i$, a *start* i *cilj* tada predstavljaju konkavne točke S tipa. Prema tome, putanja $\mathcal{P}_W$ je najkraća putanja među svim mogućim putanjama kojima su lomne točke u središtima polja mrežaste karte zauzeća.

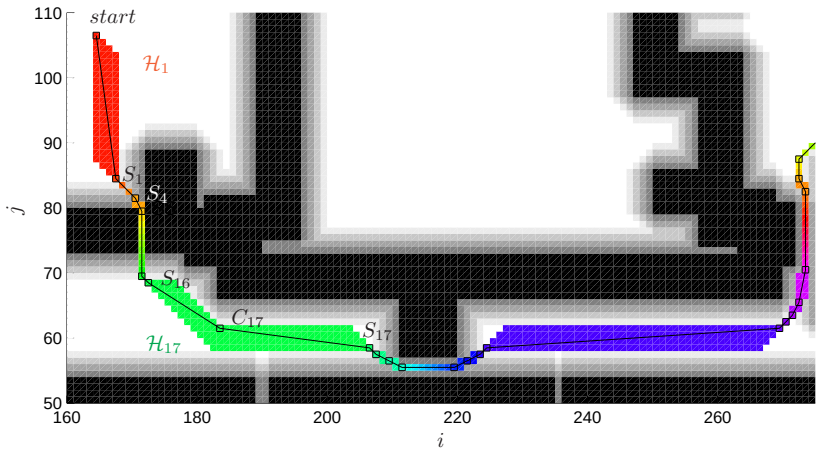
Povećanjem razlučivosti mreže, odnosno smanjenjem veličine polja, područje skupa $\mathcal{F}$ neće znatno promijeniti svoj oblik budući da graf stvoren iz mrežaste karte zauzeća i dalje sadrži optimalne putanje sastavljene od ravnih i dijagonalnih prijelaza. Međutim, većom razlučivošću prepreke će biti vjernije opisane i područje skupa $\mathcal{F}$ približit će se stvarnim preprekama, a konkavne točke približit će se rubovima prepreka. Kao rezultat

i s obzirom na već pokazano da je putanja $\mathcal{P}_W$ najkraća putanja uz uvjet da su lomne točke u središtima polja mrežaste karte zauzeća, povećanjem razlučivosti mreže putanja $\mathcal{P}_W$ bit će nalik najkraćoj Euklidskoj putanji. □

2.4.2 Algoritam DD* za izračunavanje najkraće putanje u grafovima s proizvoljnim težinama

Prethodno opisana putanja $\mathcal{P}_W$ ima dva nedostatka: (a) najkraća putanja $\mathcal{P}_W$ prolazi blizu samih vrhova prepreka što je nepoželjno u planiranju gibanja robota, a korištenje sigurnosne maske cijena nije moguće budući da Witkowskijev algoritam nije primijenjiv u grafovima s proizvoljnim težinama; (b) izračun koraka (1) i (2) za cijelu putanju od starta do cilja je memorijski i vremenski vrlo zahtjevan, pogotovo u dinamičkim prostorima u čijem se slučaju ovi koraci moraju ponovno izvršiti pri svakoj promjeni u prostoru mobilnog robota.

Ova dva nedostatka riješena su modifikacijom Witkowskijeva algoritma tako što se umjesto dva prolaza algoritma BFS koriste dva prolaza algoritma D*, jedan izvršen od cilja do starta, standardni algoritam D*, a drugi izvršen od starta do cilja, tzv. reverzni algoritam D* (RD*). Algoritam D* koristi skup $\mathcal{O}$ u koji dodaje one čvorove koje trenutno razmatra. Svakom proširenom čvoru n određuje vrijednosti $g(n)$, $k(n)$ za kasnije procese dinamičkog planiranja i pokazivače $b(n)$ za rekonstrukciju optimalne putanje do cilja. Algoritam RD* koristi skup $\mathcal{O}_R$ u koji dodaje one čvorove koje trenutno razmatra. Svakom proširenom čvoru n određuje vrijednosti $h(n)$, $k_R(n)$ za kasnije procese dinamičkog planiranja i pokazivače $b_R(n)$ za rekonstrukciju optimalne putanje do starta. Algoritam RD* dan je istim algoritmom 2.5, osim što umjesto vrijednosti g, k, b i skupa $\mathcal{O}$ pamti h, k_R, b_R i skup $\mathcal{O}_R$ respektivno. Adekvatno je izmijenjen i algoritam 2.4 koji dodaje čvor u skup $\mathcal{O}$ i njega nazivamo **dodaj-RD*** (n, h_{min}).



Slika 2.13. *Skupovi čvorova $\mathcal{H}_i$, konkavne točke tipa C i S i linijski segmenti koji ih spajaju određeni na mrežastoj karti zauzeća sa sigurnosnom maskom cijena.*

U inicijalnom izvođenju algoritma DD*, izvršava se iscrpno pretraživanje oba algoritma D* i RD*. Time su određene cijene $g(n)$ i $h(n)$ za svaki čvor n u grafu $\mathcal{G}$. Ukupna cijena putanje $f(n) = g(n) + h(n)$ je cijena optimalne putanje od starta do cilja koja prolazi kroz čvor n i jednaka je onoj dobivenoj Witkowskijevim algoritmom. Stoga se može odrediti i skup $\mathcal{F}$, a putanja $\mathcal{P}_W$ određuje se prema prethodno opisanim koracima (1) i (2) od starta do cilja i za nju vrijedi da je najkraća putanja u geometrijskom prostoru s tako proširenim preprekama uz uvjet da su lomne točke u središtima polja (teorem 2.5). Primjer područja skupa $\mathcal{F}$ podijeljenoga na podskupove $\mathcal{H}_i$ i konkavne točke određene na mrežastoj karti zauzeća sa sigurnosnom maskom cijena prikazani su na slici 2.13.

2.4.3 Dinamičko planiranje algoritmom DD*

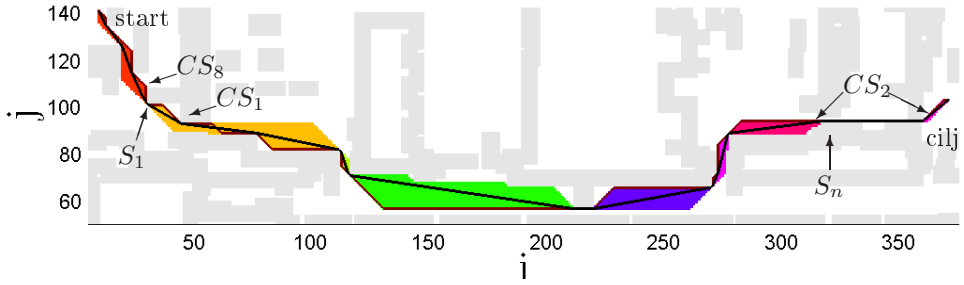
Promjena u prostoru pokreće proces dinamičkog planiranja. Prvo se dinamički planira algoritmom D*, a nakon toga algoritmom RD*. Ovaj je slijed nužan budući da je uvjet zaustavljanja algoritma RD* određen tek nakon završetka procesa dinamičkog planiranja algoritmom D*.

Dinamičko planiranje algoritmom D* i određivanje S točke

Čvorovi se redom pretražuju od novo promijenjenih čvorova dokle god postoji čvor u skupu $\mathcal{O}$ kojem je cijena k manja nego cijena g čvora R koji označava trenutnu poziciju robota. Ovaj uvjet zovemo uvjet zaustavljanja algoritma D* u procesu dinamičkog planiranja. Trenutni čvor R određuje granicu cijene do koje se pretražuju čvorovi. Tada je slijeđenjem pokazivača iz svakog čvora na putanji počevši od čvora R određena nova optimalna putanja $\mathcal{P}_{D^*}$. Duž nove putanje $\mathcal{P}_{D^*}$, traži se robotu najbliža konkavna točka S tipa, označena sa S_R, koja leži također i na staroj putanji $\mathcal{P}_{D^*}$ prije promjene u prostoru. Točka S_R odredit će uvjet zaustavljanja algoritma RD*. Točka S_R u algoritmu RD* ima ulogu trenutnog čvora kao što je čvor R u algoritmu D* i određuje granicu cijene do koje se pretražuju čvorovi. Točka S_R se odabire zato da se smanji izračun nove putanje $\mathcal{P}_W$. Umjesto da se uzima ciljni čvor G kao trenutni čvor za algoritam RD* i računa nova putanja $\mathcal{P}_W$ skroz do cilja G, pronalaženjem S točke na novoj izračunatoj putanji $\mathcal{P}_{D^*}$ koja je ujedno na staroj putanji $\mathcal{P}_W$ potrebno je izračunati samo dio nove putanje $\mathcal{P}_W$ između čvora R i čvora S_R. Dio putanje $\mathcal{P}_W(R, S_R)$ i ostatak stare putanje $\mathcal{P}_W(S_R, G)$ zajedno čine putanju $\mathcal{P}_W(R, G)$ kakvu bi odredio inicijalni izračun algoritma DD* od čvora R do G. To osigurava svojstvo S točke kroz koju mora proći samo jedna optimalna putanja pa tako i putanja $\mathcal{P}_W$. Određivanje S točke na putanji $\mathcal{P}_{D^*}$ prije nego je izvršen algoritam RD* i određen skup $\mathcal{F}$ je trikovit dio ovog algoritma. Određivanje S točke na putanji $\mathcal{P}_{D^*}$ opisat ćemo najprije za binarnu mrežastu kartu zauzeća, a nakon toga, uz malu izmjenu određenih uvjeta, postupak će vrijediti i za mrežastu kartu zauzeća sa sigurnosnom maskom cijena.

Pogledajmo pozicije konkavnih točaka S tipa na primjeru prikazanom na slici 2.14,

gdje su iscrtane putanje $\mathcal{P}_W$ i $\mathcal{P}_{D^*}$ i područje skupa $\mathcal{F}$. S točke se nalaze između dvaju slijednih skupova $\mathcal{H}_i$ i $\mathcal{H}_j$, $i < j \leq H$, koji sadrže više od dva čvora. Neka je $\mathcal{H}_i$ sastavljen od konveksnih skupova tipa CS_k, a $\mathcal{H}_j$ od konveksnih skupova tipa CS_l, gdje su $k, l \in \{1, \ldots, 8\}$. Postoje dva slučaja lokacije S točke: (1) između međusobno različitih tipova konveksnih skupova CS_k i CS_l, $l \neq k$ i (2) između konveksnih skupova CS_k i CS_l istog tipa, $k = l$, ako se prepreke nalaze s obje strane putanje između skupova $\mathcal{H}_i$ i $\mathcal{H}_j$. Budući da se putanja $\mathcal{P}_{D^*}$ također nalazi unutar skupa $\mathcal{F}$, moguće je odrediti

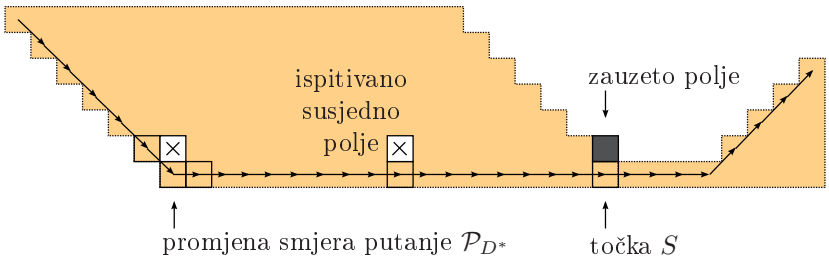


Slika 2.14. *Pozicije konkavnih točaka S tipa s obzirom na konveksne skupove CS_k u skupu $\mathcal{F}$.*

analiziranjem prijelaza na putanji kojem tipu konveksnog skupa CS_k, $k \in \{1, \ldots, 8\}$, pripada određeni dio putanje. Također je moguće odrediti promatranjem okoline svakog polja na putanji $\mathcal{P}_{D^*}$ koja se polja putanje nalaze blizu rubova prepreka. Prema tome, moguće je samo iz putanje $\mathcal{P}_{D^*}$ i pozicija prepreka u karti odrediti poziciju S točaka. U nastavku je opisan postupak određivanja pozicije S točke s obzirom na spomenute slučajeve (1) i (2).

Niz uzastopnih istovrsnih prijelaza na putanji nazivamo linijskim segmentom putanje. Tip konveksnog skupa CS_k, $k \in \{1, \ldots, 8\}$, kojemu pripada dio putanje $\mathcal{P}_{D^*}$ otkriven je promatranjem orijentacija dvaju uzastopnih linijskih segmenata putanje. Orijentacija trećeg linijskog segmenta putanje u odnosu na prva dva segmenta određuje nalazi li se treći segment i dalje u konveksnom skupu istog tipa CS_k. Ako se orijentacija trećeg linijskog segmenta putanje ne podudara s orijentacijom prvog segmenta tada se treći segment nalazi u drugom konveksnom skupu CS_l, $l \neq k$, i tada postoji konkavna točka S tipa u prethodnom (drugom) linijskom segmentu putanje (slučaj (1)).

Postupak određivanja S točke na putanji $\mathcal{P}_{D^*}$ za slučaj (1) prikazan je na slici 2.15. Kada se otkrije polje putanje u kojem počinje drugi linijski segment putanje (promjena smjera putanje $\mathcal{P}_{D^*}$), određuje se susjedno polje označeno sa x s konveksne strane kuta kojeg formiraju prvi i drugi linijski segment putanje u tom polju. Treći linijski segment ima različitu orijentaciju od prvog linijskog segmenta i odgovarajuće susjedno polje označeno sa x (lijevi susjed svakom polju na putanji) mora se ispitati od svakog polja na prethodnom (drugom) linijskom segmentu putanje. Prvo polje putanje na drugom linijskom segmentu čiji je provjeravani susjed zauzeto polje jest S točka. Primjer slučaja (1) može se vidjeti na slici 2.14: točka S_1 detektirana je između konveksnih skupova tipa CS_8 i CS_1.

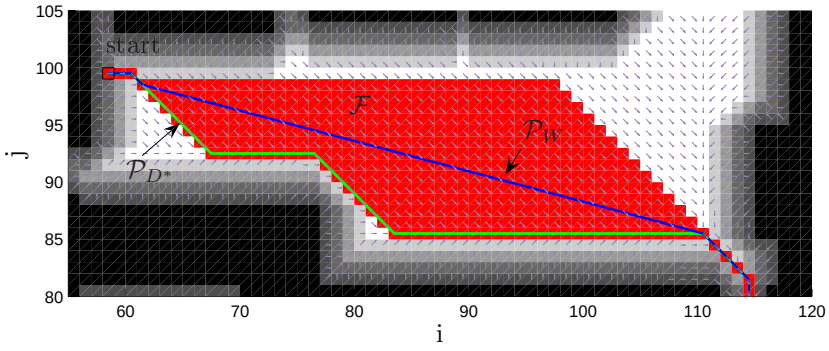


Slika 2.15. *Određivanje točke S na putanji $\mathcal{P}_{D^*}$: slučaj (1).*

Postupak određivanja S točke na putanji $\mathcal{P}_{D^*}$ za slučaj (2) određuje se uzastopnom provjerom postoji li duž jednog linijskog segmenta prepreka s obje strane putanje. Točka S je prva točka blizu ruba prepreke na tom linijskom segmentu. Ovaj slučaj se može vidjeti na slici 2.14: točka S_n detektirana je između dvaju CS_2 konveksnih skupova.

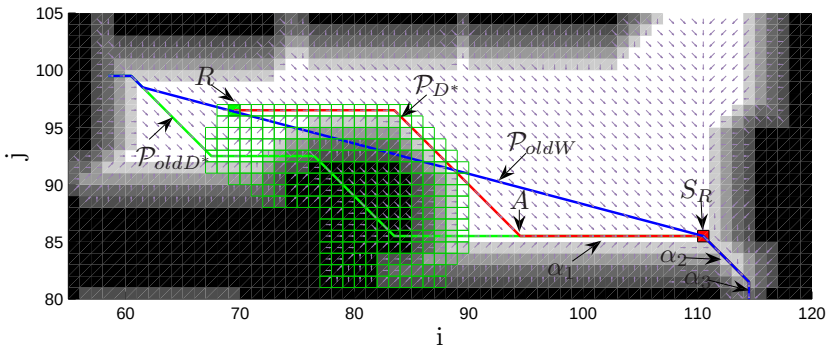
Određivanje točke S na putanji $\mathcal{P}_{D^*}$ određenoj u grafu stvorenom iz mrežaste karte zauzeća sa sigurnosnom maskom cijena je slično opisanom postupku uz malu izmjenu uvjeta koji se provjerava. Može se primijetiti da područje skupa $\mathcal{F}$ proračunato na ovakvom grafu prolazi kroz slobodna polja izvan sigurnosne maske (ako je moguće) kao što je prikazano u primjeru na slici 2.13. Konkavne točke C tipa blizu su proširenih prepreka za sigurnosnu masku cijene. Područje skupa $\mathcal{F}$ širine je jednog polja (S točke) ako su susjedna polja unutar sigurnosne maske, dakle vrijednost zauzeća tih polja je veća od jedan, što je slučaj u uskim prolazima. Uvjet da je točka S blizu zauzetog polja (kao na slici 2.15), ovdje je transformiran u uvjet da je točka S blizu sigurnosne maske, odnosno blizu polja kojem je vrijednost zauzeća veća od jedan.

Određivanje točke S_R u algoritmu DD* opisano je primjerima prikazanim na slikama 2.16 koja prikazuje inicijalno pretraživanje i 2.17 koja prikazuje dinamičko planiranje algoritmom D* i detekciju točke S_R. Radi jasnoće, prikazan je samo dio prostora u okolici startne pozicije mobilnog robota.



Slika 2.16. *Inicijalno pretraživanje: pokazivači algoritma D*, putanja $\mathcal{P}_{D^*}$, područje skupa $\mathcal{F}$ algoritma DD* i putanja $\mathcal{P}_W$.*

U inicijalnom pretraživanju određeno je područje skupa $\mathcal{F}$ i putanja $\mathcal{P}_W$. Naznačeni su pokazivači algoritma D* koji određuju optimalne putanje iz svakog čvora do cilja i



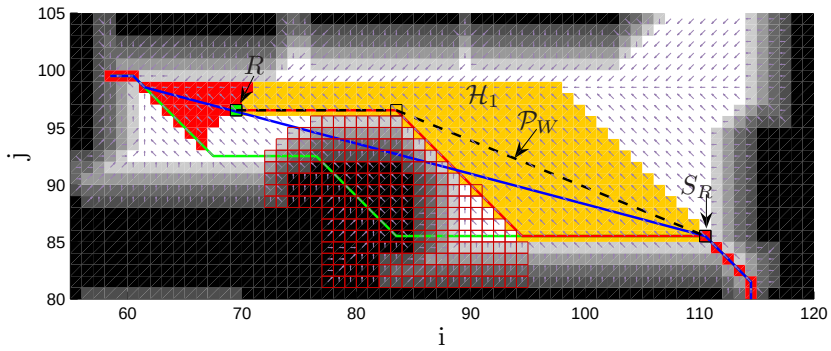
Slika 2.17. *Dinamičko planiranje algoritmom D*: pokazivači algoritma D*, □ – svi pretraženi čvorovi, nova putanja $\mathcal{P}_{D^*}$, određena točka S_R.*

optimalna putanja $\mathcal{P}_{D^*}$ algoritma D*. Robot prati inicijalnu putanju $\mathcal{P}_W$ (označenu s $\mathcal{P}_{oldW}$ na slici 2.17) i kada stigne u poziciju označenu s R otkrije novu prepreku prikazanu skupinom crnih polja. Nova je prepreka proširena uzimajući u obzir dimenzije robota i sigurnosnu masku cijena izračunatu oko nje. Algoritmom D* pretraženi su čvorovi označeni kvadratima. Broj pretraženih čvorova određuje čvor R svojom cijenom (pretražuju se svi čvorovi kojima je cijena k manja od cijene $g(R)$). Tim su čvorovima određene nove optimalne putanje do cilja i cijene g. Nova putanja $\mathcal{P}_{D^*}$ dobivena je praćenjem pokazivača iz čvora R do cilja. Točka S_R tražena je na novoj putanji $\mathcal{P}_{D^*}$ od točke od koje se nova i stara putanja $\mathcal{P}_{D^*}$ preklapaju (stara putanja $\mathcal{P}_{D^*}$ je označena s $\mathcal{P}_{oldD^*}$, a početna točka preklapanja je označena s A na slici 2.17). Orijentacije linijskih segmenata putanje razmatraju se od točke A i označeni su redom s α_1, α_2 i α_3. Orijentacija trećeg linijskog segmenta nije jednaka orijentaciji prvog segmenta (slučaj (1)) te se S točka mora nalaziti negdje na drugom linijskom segmentu. Susjedno polje s konveksne strane kuta kojeg formiraju prvi i drugi linijski segment provjerava se na svakom polju u drugom linijskom segmentu putanje (desni susjed svakom polju drugog segmenta putanje). Točka S_R je prvi čvor čiji provjeravani susjed ima vrijednost zauzeća veću od jedan.

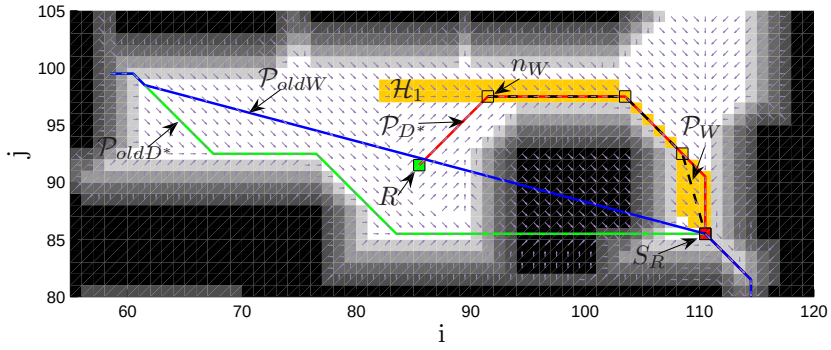
Dinamičko planiranje algoritmom RD*

Određena točka S_R postavljena je kao trenutni čvor za algoritam RD*. Čvorovi se pretražuju od novo promijenjenih čvorova prema trenutnom čvoru, S_R, sve dok nije cijena h trenutnog čvora manja od cijene k_R najboljeg čvora u skupu $\mathcal{O}_R$. Tada se računanjem $f(n)$ određuje novo područje skupa $\mathcal{F}$ od čvora R do točke S_R, a zatim se određuje putanja $\mathcal{P}_W(R, S_R)$ između tih dviju točaka prema prethodno opisanoj metodi određivanja putanje $\mathcal{P}_W$. Skupovi $\mathcal{H}_i$, $i = 1, \ldots, H$ određuju se prema algoritmu 2.10, od čvora R do čvora S_R.

Ovaj korak procesa dinamičkog planiranja prikazan je na slici 2.18, a nastavlja se na primjer prikazan slikama 2.16 i 2.17. Algoritmom RD* pretraženi su čvorovi oz-



Slika 2.18. *Dinamičko planiranje algoritmom RD*: pokazivači algoritma RD*, □ – svi pretraženi čvorovi, područje skupa $\mathcal{H}_1$ algoritma DD* i nova putanja $\mathcal{P}_W(R, S_R)$.*



Slika 2.19. *Dinamičko planiranje: trenutna pozicija robota je izvan područja skupa $\mathcal{F}$.*

načeni kvadratima. Broj pretraženih čvorova određuje čvor S_R tako što su pretraženi svi čvorovi kojima je cijena k_R manja od cijene $h(S_R)$. Tim su čvorovima određene nove optimalne putanje do starta i cijene h. Prikazani su pokazivači algoritma RD* koji određuju optimalnu putanju iz svakog čvora do starta. Po završetku algoritma RD* formirano je područje skupa $\mathcal{F}$ prikazano ispunjenim kvadratima. Skup $\mathcal{H}_1$ formiran je od čvora R prema algoritmu 2.10. Novi skup $\mathcal{H}_1$ prikazan je svjetlijom sivom (narančastom) bojom ispunjenim kvadratima. Nova putanja $\mathcal{P}_W(R, S_R)$ određena je od čvora R do točke S_R. Ukupna putanja od pozicije robota do cilja određena je spajanjem nove putanje $\mathcal{P}_W(R, S_R)$ od čvora R do točke S_R, i stare putanje $\mathcal{P}_W(S_R, G)$ od točke S_R do cilja. Ova ukupna putanja ekvivalentna je putanji $\mathcal{P}_W(R, G)$ izračunatoj kao u inicijalnom koraku iz čvora R do cilja što je osigurano pronalaskom S točke na putanji $\mathcal{P}_{D^*}$.

Prethodni primjer prikazuje specijalan slučaj u kojem je čvor R i prije i poslije promjene u prostoru usred područja skupa $\mathcal{F}$. Međutim, trenutna pozicija robota može biti u nekom čvoru R, koji je prije promjene u prostoru u području skupa $\mathcal{F}$, ali promjena u prostoru može biti takva da nakon procesa dinamičkog planiranja algoritmima D* i RD* čvor R više nije u području novog skupa $\mathcal{F}$. Razlog tomu je što je vrijednost f optimalna s obzirom na start i cilj, a ne trenutnu poziciju robota i cilj. Općeniti slučaj

dinamičkog planiranja prikazan je na slici 2.19, a nastavlja se na inicijalni izračun prikazan na slici 2.16. Robot prati inicijalnu putanju $\mathcal{P}_W$ (označenu s $\mathcal{P}_{oldW}$) i kad stigne u poziciju označenu s R otkrije novu prepreku. Dinamičko planiranje algoritmom D* je prema čvoru R, a dinamičko planiranje algoritmom RD* je prema točki S_R. Naznačeni su pokazivači algoritma D* i nova optimalna putanja $\mathcal{P}_{D^*}$ dobivenom praćenjem pokazivača iz čvora R. Čvor R nalazi se izvan područja skupa $\mathcal{F}$ koje je prikazano ispunjenim kvadratima. Ako se dinamičko planiranje algoritmom D* proširi skroz do starta, a dinamičko planiranje algoritmom RD* skroz do cilja, područje skupa $\mathcal{F}$ protezat će se od starta do cilja, ali čvor R i dalje neće biti unutar područja. Potrebno je povezati čvor R s područjem skupa $\mathcal{F}$ bez gubitka optimalnosti. Budući da je algoritam D* odredio novu optimalnu putanju $\mathcal{P}_{D^*}$ iz čvora R, ovaj se problem rješava nalaženjem najbližeg čvora na putanji $\mathcal{P}_{D^*}$ označenim s n_W koji je u skupu $\mathcal{F}$. Čvor n_W je

Algoritam 2.11: gibanje-robota-DD*($start, cilj$)

$\forall n \in \mathcal{N}$, $k(n) \leftarrow \infty$, $g(n) \leftarrow \infty$, $b(n) \leftarrow n$
$k_R(n) \leftarrow \infty$, $h(n) \leftarrow \infty$, $b_R(n) \leftarrow n$
$R \leftarrow start$, $\mathcal{O} \leftarrow \emptyset$, $\mathcal{O}_R \leftarrow \emptyset$ *//Inicijalizacija algoritama D* i RD**
$\mathcal{P}_W \leftarrow \emptyset$ *//Inicijalizacija najkraće putanje*
dodaj-D*($cilj, 0$)
dodaj-RD*($start, 0$)
dok $R \neq cilj$
  $\mathcal{I}_R \leftarrow$**promijenjeni-čvorovi**(R)
  ako $R = start$ ili $\mathcal{I}_R \setminus \mathcal{O} \neq \emptyset$ *//Inicijalno ili dinamičko planiranje*
    D*($\mathcal{I}_R, R$)
    $i \leftarrow 1$, $\mathcal{P}_{D^*}[i] \leftarrow R$
    dok $\mathcal{P}_{D^*}[i] \neq cilj$
      $i \leftarrow i + 1$
      $\mathcal{P}_{D^*}[i] \leftarrow b(\mathcal{P}_{D^*}[i-1])$
    kraj
    $S_R \leftarrow$**odredi-S-točku**($\mathcal{P}_W, \mathcal{P}_{D^*}$) *//vraća cilj ako je* $\mathcal{P}_W = \emptyset$
    RD*($\mathcal{I}_R, S_R$)
    $\mathcal{F} \leftarrow \{n \mid g(n) + h(n) = g(S_R) + h(S_R)\}$
    $i \leftarrow 1$
    dok $\mathcal{P}_{D^*}[i] \notin \mathcal{F}$
      $i \leftarrow i + 1$
    kraj
    $n_W \leftarrow \mathcal{P}_{D^*}[i]$
    određivanje-podskupova-optimalnih-putanja($n_W, S_R, \mathcal{F}$)
    za $\forall \mathcal{H}_i$, $i = 1, \ldots, H$
      $\mathcal{P}_i \leftarrow$**najkraća-putanja**($S_{i-1}, S_i, \mathcal{H}_i$)
    kraj
    $\mathcal{P}'_W \leftarrow \mathcal{P}_1 \cup \cdots \cup \mathcal{P}_H$
    $\mathcal{P}_W \leftarrow \mathcal{P}_{D^*}(R, n_W) \cup \mathcal{P}'_W \cup \mathcal{P}_W(S_R, G)$
  kraj
  $R \leftarrow$**prati-putanju-W**($\mathcal{P}_W$)
kraj

početni čvor za određivanje nove putanje $\mathcal{P}_W$. Skupovi $\mathcal{H}_i$, $i = 1, \ldots, H$, formiraju se od čvora n_W do S_R. Ukupna putanja od pozicije robota do cilja određena je spajanjem putanje $\mathcal{P}_{D^*}(R, n_W)$ od trenutne pozicije robota (čvora R) do čvora n_W, nove putanje $\mathcal{P}_W(n_W, S_R)$ od čvora n_W do točke S_R, i stare putanje $\mathcal{P}_W(S_R, G)$ od točke S_R do cilja.

Pseudokod algoritma koji tijekom gibanja robota od starta do cilja poziva algoritam DD* dan je algoritmom 2.11. Algoritam DD* izvršava se inicijalno, ako je $R = start$, te u slučaju promjena u prostoru koje je robot otkrio senzorom. Funkcija promijenjeni-čvorovi(R) u liniji 7 vraća sve čvorove kojima se promijenila vrijednost zauzeća i njih dalje obrađuju algoritmi D* i RD*. Po završetku algoritma D* nova optimalna putanja $\mathcal{P}_{D^*}$ određuje se pomoću pokazivača u linijama 10-14. Na novoj putanji $\mathcal{P}_{D^*}$ određuje se točka S_R funkcijom **odredi-S-točku** čiji su detalji prethodno opisani. Dalje se algoritmom RD* određuju potrebne cijene h da bi se odredio skup $\mathcal{F}$. Točka n_W iz koje se računaju skupovi $\mathcal{H}_i$ određuje se u linijama 18-22. Određeni skupovi $\mathcal{H}_i$, $i = 1, \ldots, H$ i konkavne točke S i C tipa tretiraju se kao skrivene varijable, a koriste u stvaranju najkraćih putanja $\mathcal{P}_i$ unutar svakog skupa $\mathcal{H}_i$ (linije 24-27). Ukupna putanja $\mathcal{P}_W$ određena je kao unija pojedinačnih putanja određenih prethodno. Tu putanju robot slijedi do cilja, a ponovno se proračunava u slučaju promjene u prostoru.

2.4.4 Eksperimentalni rezultati

Rezultati dvaju eksperimenata na robotu Pioneer 3DX opremljenim laserskim senzorom udaljenosti prikazani su radi validacije algoritma DD*. Prvi eksperiment prikazuje slučaj navigacije robota u potpuno poznatom prostoru, a drugi eksperiment prikazuje slučaj navigacije robota u istom prostoru, ali s postavljenim nepoznatim preprekama. Korišten je CAD model prostora mobilnog robota sastavljen od mnoštva linija koje predstavljaju prepreke. Iz takvog modela izgrađuje se mrežasta karta zauzeća tako što se prostor mrežasto podijeli na konačan broj polja, a linije se pretvaraju u diskretni niz točaka. Svakoj točki linije određuje se polje kojem točka pripada kako slijedi. Točka čije su koordinate $(x, y) \in \mathbb{R}^2$ pretvara se u cjelobrojne koordinate mrežaste karte zauzeća $(i_n, j_n) \in \mathbb{N}^2$ prema sljedećem izrazu:

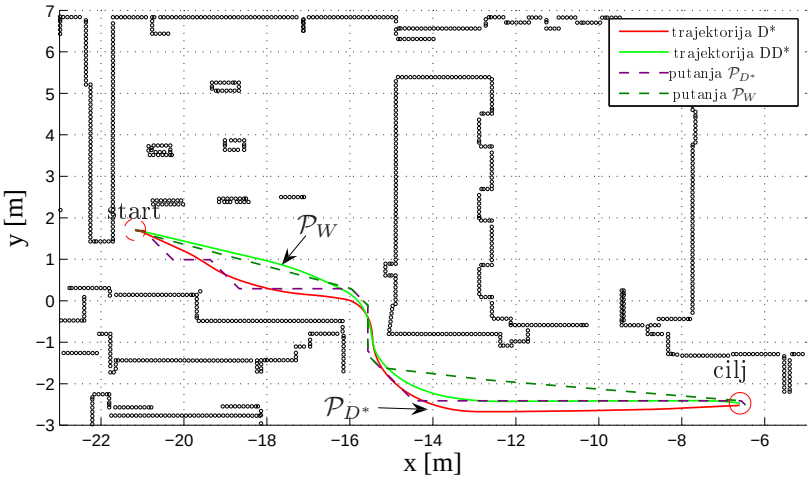
$$(i_n, j_n) := (\lceil \frac{x - x_0}{e_{polje}} \rceil, \lceil \frac{y - y_0}{e_{polje}} \rceil), \tag{2.23}$$

gdje je e_{polje} duljina stranice polja, a x_0 i y_0 najmanje koordinate prostora (ishodište je lijevi donji ugao karte prostora). Prema tome, polje $n \in \mathcal{K}$ koje ima cjelobrojne koordinate centra u $(i_n, j_n) \in \mathbb{N}^2$ postaje zauzeto. Nepoznate prepreke upisuju se u mrežastu kartu zauzeća kada ih robot otkrije svojim laserskim senzorom udaljenosti. Laserski senzor udaljenosti daje niz točaka u prostoru u kojima je zraka lasera udarila u prepreku. Za svaku točku određuje se prema (2.23) u koje polje upada i to polje postaje zauzeto. Okolnja polja ažuriraju vrijednost zauzeća da bi prepreke bile proširene za dimenzije robota. Dodatno se oko zauzetih polja unutar sigurnosne maske cijena računaju vrijednosti zauzeća prema izrazu (2.1). Dinamičke promjene u prostoru otkrivaju

se laserskim senzorom tako što se u fiksnim vremenskim intervalima (svakih 100 ms) snima stanje prostora laserom i ažurira mrežasta karta zauzeća. Robot je lokaliziran AMCL lokalizacijom (engl. *adaptive Monte Carlo localization*), koja koristi opisanu kartu i laserska očitanja. Veličina prostora korištena u eksperimentima je 539 x 164 polja, gdje je veličina polja $e_{polje} = 0.1$ m.

Razvijeni algoritam DD* uspoređen je sa standardnim algoritmom D* s obzirom na brzinu slijeđenja putanje i vrijeme izvođenja. Trajektorije robota i profili brzina dobiveni objedinjenjem algoritma DD* s algoritmom dinamičkog prozora uspoređeni su istima dobivenim objedinjenjem algoritma D* s algoritmom dinamičkog prozora[4]. Obje putanje $\mathcal{P}_W$ i $\mathcal{P}_{D^*}$ određene su na grafu stvorenom iz mrežaste karte zauzeća sa sigurnosnom maskom cijena.

Prvi eksperiment



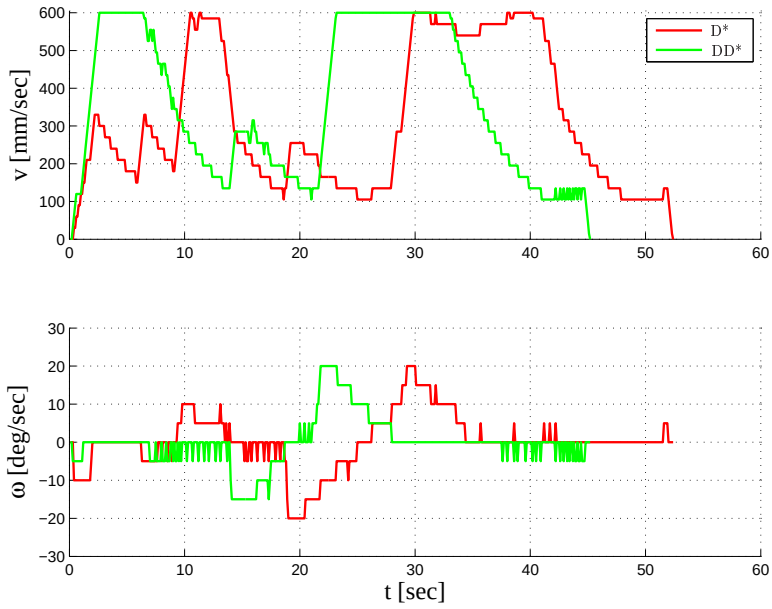
Slika 2.20. *Trajektorija robota prateći putanju $\mathcal{P}_{D^*}$ i trajektorija robota prateći putanju $\mathcal{P}_W$, obje izračunate na mrežastoj karti zauzeća sa sigurnosnom maskom cijena.*

Tablica 2.2. *Rezultati prvog eksperimenta.*

Algoritam	Duljina putanje [m]	Duljina trajektorije [m]	vrijeme gibanja [s]
D*	17.14	16.37	52.4
DD*	16.48	16.18	45.2

Slika 2.20 prikazuje trajektorije, a slika 2.21 profile brzina dobivene u prvom eksperimentu prateći putanju $\mathcal{P}_{D^*}$ i $\mathcal{P}_W$, respektivno. Kao što se vidi iz tablice 2.2 i slike

[4]Objedinjavanje algoritama planiranja putanje s algoritmom dinamičkog prozora opisano je u poglavlju 5.



Slika 2.21. *Profili brzina prvog eksperimenta.*

2.21, algoritam DD* proizvodi kraću putanju i trajektoriju robota nego algoritam D* i robot dostiže cilj 13.7% brže.

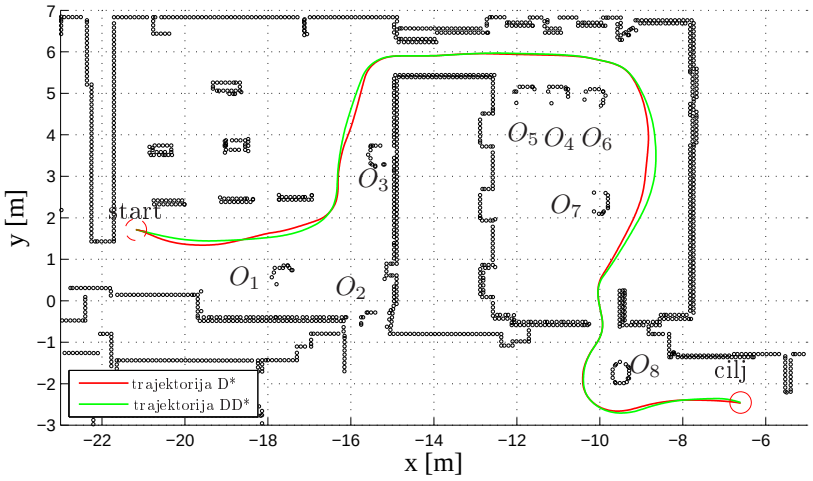
Drugi eksperiment

U drugom eksperimentu koristili smo isti prostor, ali smo postavili osam prepreka (O_1-O_8) radi testiranja sposobnosti predloženog algoritma u zaobilaženju prepreka. Slika 2.22 prikazuje trajektorije, a slika 2.23 profile brzina dobivene u ovom eksperimentu prateći putanju $\mathcal{P}_{D^*}$ i $\mathcal{P}_W$, respektivno. Kao što se vidi u tablici 2.3, algoritam DD* proizvodi kraću putanju i dulju trajektoriju nego algoritam D*, a robot dostiže cilj 13.2% brže.

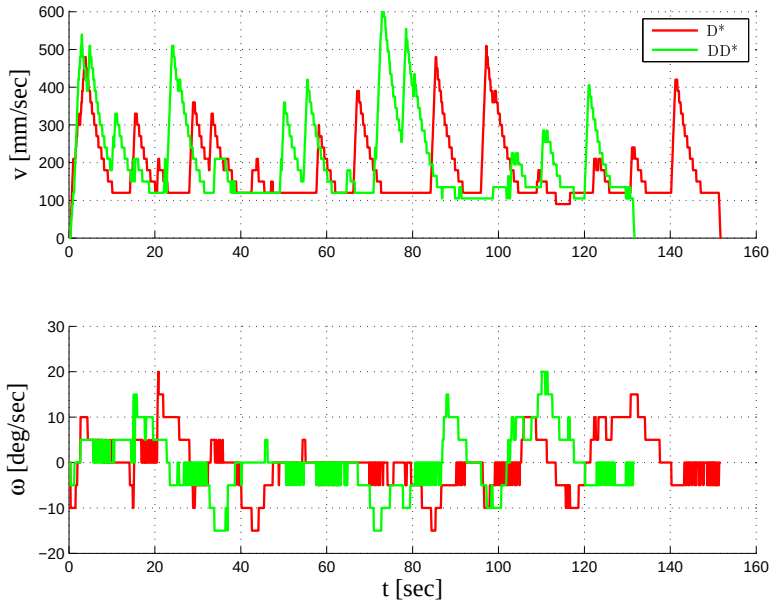
Tablica 2.3. *Rezultati drugog eksperimenta.*

Algoritam	Duljina putanje [m]	Duljina trajektorije [m]	Vrijeme gibanja [s]
D*	29.40	27.06	151.7
DD*	29.09	27.26	131.7

Tablica 2.4 prikazuje vrijeme inicijalnog planiranja kao i vremena dinamičkog planiranja (označena kao t_p) i broj pretraženih čvorova (označenih kao E) pri detekciji svih osam prepreka, što jasno indicira da je algoritam DD* računski zahtjevniji nego algoritam D*. Ti su rezultati očekivani budući da dinamičko planiranje algoritmom DD*



Slika 2.22. *Trajektorija robota prateći putanju $\mathcal{P}_{D^*}$ i trajektorija robota prateći putanju $\mathcal{P}_W$, obje izračunate na mrežastoj karti zauzeća sa sigurnosnom maskom cijena.*

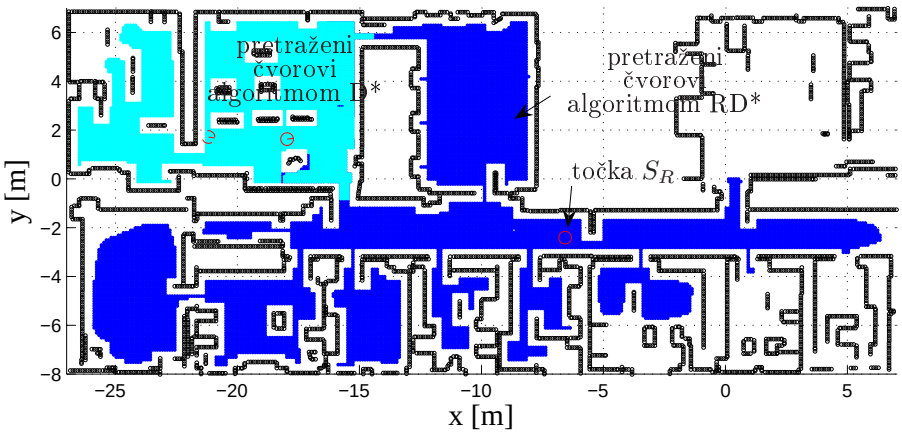


Slika 2.23. *Profili brzina u drugom eksperimentu.*

uključuje dinamičko planiranje algoritma D* praćeno algoritmom RD* i računanjem najkraće putanje $\mathcal{P}_W$ kroz područje skupa $\mathcal{F}$. Scenarij najgoreg slučaja pojavljuje se u zaobilaženju prepreke O_2 zbog njenog položaja kod vrata pa ju robot ne može jednostavno zaobići. Umjesto toga robot mora ići putanjom kroz druga vrata sobe. Pretraženi čvorovi prikazani su na slici 2.24. Veliki broj pretraženih čvorova i s tim u skladu dugi izračun algoritma RD* uzrokovani su pozicijom točke S_R, koja je daleko od prepreke O_2, tj. skoro kod cilja.

Tablica 2.4. *Vremena planiranja i broj pretraženih čvorova algoritama D* i DD**

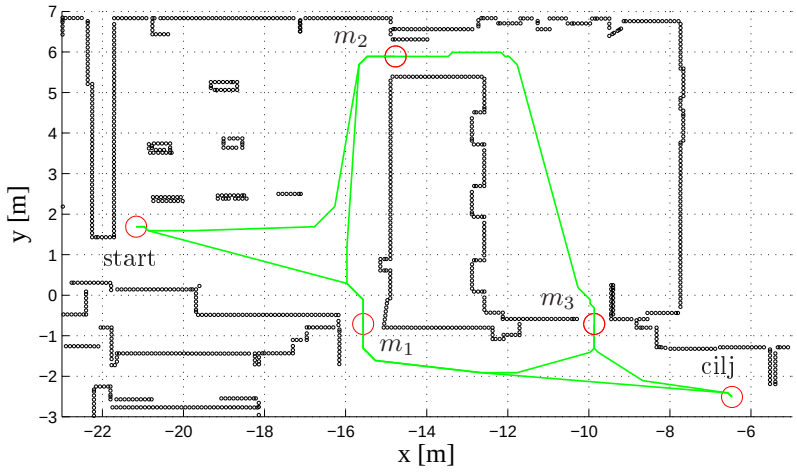
Algoritam	D* t_p [ms] (E)	RD* t_p [ms] (E)	$\mathcal{P}_W$ t_p [ms]	DD* t_p [ms]
Inicijalno	206(42420)	200(42485)	1	407
O_1	8(1437)	3(823)	1	12
O_2	48(9392)	91(17533)	3	142
O_3	2(143)	0(247)	1	3
O_4	6(1056)	23(6396)	2	31
O_5	2(381)	0(157)	0	2
O_6	7(1126)	11(3860)	0	18
O_7	6(1248)	2(846)	1	9
O_8	11(2783)	27(7602)	0	38



Slika 2.24. *Čvorovi pretraženi algoritmima D* i RD* kod najgoreg slučaja dinamičkog planiranja.*

Vrijeme dinamičkog planiranja moguće je ograničiti hijerarhijskom reorganizacijom grafa pretraživanja, što omogućava hijerarhijsko planiranje putanje. Prikladno rješenje je hijerarhijska dekompozicija prostora zasnovana na hijerarhijskim grafovima (H-Grafovi; [33, 34]). Hijerarhijska apstrakcija može reducirati probleme eksponencijalne složenosti na linearnu [66]. U radu [114] razvijeno je optimalno hijerarhijsko planiranje kao proširenje Cagigasovog hijerarhijskog algoritma D* [18]. Optimalnost globalne putanje dobivena je optimalnim postavljanjem tzv. mosnih čvorova potrebnih za stvaranje hijerarhije. Skup optimalnih unaprijed izračunatih parcijalnih putanja pohranjen je u mosnim čvorovima. Parcijalne putanje definiraju povezanost i cijenu između mosnih čvorova. skup parcijalnih putanja osigurava brzo dinamičko planiranje tako što se pretražuju samo mosni čvorovi na višoj razini hijerarhije. Mosni čvorovi postavljeni su na

sredinu prolaza kod vrata, kao što je prikazano na slici 2.25. Slika 2.25 također prikazuje



Slika 2.25. *Skup parcijalnih putanja između mosnih čvorova.*

skup parcijalnih putanja koji uključuje i start i cilj. Svaka parcijalna putanja izračunata je kao putanja $\mathcal{P}_W$. U procesu dinamičkog planiranja, algoritam mora pretražiti samo mosne čvorove pridružene sobi u kojoj se nalazi robot i nakon toga pretražiti nekoliko čvorova unutar sobe. Algoritam D* prvo pretražuje mosne čvorove i nakon toga dio sobe pretražujući od najboljeg mosnog čvora do trenutne pozicije robota. Naposlijetku, najbolji se mosni čvor koristi kao točka S_R za algoritam RD*. Očito je da uvođenjem hijerarhijske dekompozicije grafa veličina sobe ograničava vrijeme izračuna. U slučaju detekcije prepreke O_2, vrijeme dinamičkog planiranja se smanjilo s 142 ms na 70 ms, a broj pretraženih čvorova se smanjio s 26925 na 11644.

2.5 Sažetak

U ovom su poglavlju opisani algoritmi planiranja geometrijskih putanja robota u poznatim prostorima zasnovani na pretraživanju mrežastih karata zauzeća. Budući da je naglasak na rad u stvarnom vremenu, detaljnije su opisani algoritam A* i njegovi nasljednici. Algoritam A* pretražuje najmanji broj čvorova u grafu nužnih za jamčenje optimalne putanje. Međutim, primijenjiv je samo u statičkim prostorima. Algoritmi D* i FD* nasljednici su algoritma A*, koji imaju svojstvo dinamičkog planiranja, odnosno omogućuju promjene putanje u stvarnome vremenu pri promjenama prostora (npr. zbog novih objekata u prostoru koji nisu uključeni u kartu prostora). Algoritmi pretraživanja mrežastih karata zauzeća imaju nedostatak da je dobivena putanja sastavljena od niza povezanih linijskih segmenata s promjenama u orijentaciji segmenata koje su višekratnici od 45°. Ovaj se nedostatak odražava u potrebi za znatnim usporavanjem (često i zaustavljanjem) gibanja mobilnoga robota zbog skokovitih promjena orijentacije pri prelasku s praćenja jednog linijskog segmenta putanje na drugi. S ciljem rješavanja ovog

problema, u ovom je poglavlju razvijen algoritam planiranja putanje, nazvan dvosmjernim algoritmom D*, jer se zasniva na dva prolaza algoritma D*. Razvijeni algoritam iz mrežaste karte zauzeća izračunava najkraću putanju $\mathcal{P}_W$ u geometrijskom prostoru, čije orijentacije segmenata putanje nisu višekratnici 45° već su proizvoljne i kojime je smanjen broj skokovitih promjena orijentacije pri prelasku s jednog linijskog segmenta putanje na drugi. Algoritam nasljeđuje svojstvo dinamičkog planiranja putanje od algoritma D*. Primijenjen je na mrežastoj karti zauzeća s tzv. sigurnosnom maskom cijena oko prepreka. Sigurnosna maska odmiče putanju $\mathcal{P}_W$ dalje od prepreka povećavajući sigurnost gibanja robota. Razvijeni je algoritam provjeren eksperimentalno i uspoređen sa standardnim algoritmom D* pod istim uvjetima. Eksperimenti potvrđuju očekivane prednosti razvijenog algoritma planiranja putanje.

Poglavlje 3

Hijerarhijsko planiranje putanje

Ovo se poglavlje usredotočuje na planiranje putanje u dinamičkim prostorima s posebnim naglaskom na izvršavanje algoritma u stvarnome vremenu. Algoritmi planiranja putanje opisani u poglavlju 2 računski su prezahtjevni kada se radi o velikim prostorima. Taj se problem najčešće rješava reprezentacijom prostora mrežastom kartom nejednolike rezolucije, gdje je slobodni dio prostora predstavljen većim poljem, primjerice metodom kvadratnih stabala [110]. No putanje u takvim kartama prolaze središtima polja i nisu optimalne [110]. Višerezolucijski algoritam Field D* [30] određuje putanje kroz nejednoliku mrežastu kartu s malim vremenskim izračunom i malim utroškom memorije.

Međutim, u velikim prostorima, primjerice prostorima sastavljenima od više katova ili čak više zgrada, potrebno je napraviti hijerarhijsku apstrakciju modela prostora da bi se reducirala složenost grafa mogućih putanja. Apstrakcija se može shvatiti kao uklanjanje nebitnih detalja iz informacija organiziranih u nekoj operaciji [98]. Prikladan izbor je hijerarhijska dekompozicija prostora H-grafovima [33, 34]. Hijerarhijska apstrakcija može reducirati eksponencijalnu složenost problema planiranja na linearnu složenost [66]. Postoje razni zanimljivi pristupi nalaženju dobrih hijerarhijskih apstrakcija [51], [43], [38]. Uobičajeno se kreiranje hijerarhijske karte izvodi ručno [19], što predstavlja dugotrajan i složen posao. Općenito je problem kreiranja optimalne hijerarhijske apstrakcije kombinatorički optimizacijski problem koji nije rješiv u polinomnom vremenu [62]. Metode koje kreiraju hijerarhijske apstrakcije u polinomnom vremenu predstavljene su u [43] i [16], ali ne jamče optimalnu hijerarhiju s obzirom na izračun planiranja [3]. U [42] je pokazano da ponavljanje operacija na različitim hijerarhijskim razinama i iteriranje među razinama znatno produljuje izračun planiranja putanje. Primjerice, da bi se pronašao cilj na jednoj razini, može biti nužno poništiti ispunjen ciljni uvjet na višoj razini. To uzrokuje ponovno rješavanje međuovisnih potproblema između razina (engl. *backtracking*). Takvo rješavanje može eksponencijalno povećati vrijeme izvođenja, kao što je analizirano u [43]. Nalaženje dobre hijerarhijske apstrakcije uvodi dva dodatna ograničenja. Prvo, mora biti sačuvana konzistentnost kroz hijerarhijske razine modela [104] i drugo, model mora ostati konzistentan nakon promjena u prostoru. To može

dovesti do računski prezahtjevnih metoda. U [38] se koristi hijerarhijska reprezentacija prostora nazvana "naznačen i hijerarhijski" (engl. *annotated and hierarchical*) graf, gdje je konzistentnost zajamčena automatskim kreiranjem apstrakcijskih razina iz najdetaljnije razine. Nestrukturirane informacije u obliku oznaka sadržane su u čvorovima i lukovima grafa i koriste se u planiranju. Taj pristup koristi optimizacijski algoritam koji počinje od inicijalne hijerarhijske apstrakcije (koja može biti neoptimalna) i kontinuirano ju poboljšava kako robot stječe iskustvo. Na najdetaljnijoj razini apstrakcije koristi topološku kartu prostora. Stoga, ta metoda ovisi o automatskom kreiranju topoloških karata [89] i također o algoritmima uskupljavanja (engl. *clustering*) grafova [80].

Hijerarhijsko se planiranje putanje zasniva na procesu postupnog korigiranja putanje počevši od najviše hijerarhijske razine (visoka razina apstrakcije) prema nižim razinama (više detalja). U [19] skup optimalnih predodređenih parcijalnih putanja prvotno se pohranjuje u ključnim točkama H-grafa sa ciljem ubrzanja procesa planiranja. Taj se postupak zove *materijalizacija cijena* [56]. Materijalizacija cijena izbjegava ponovno računanje nekih dijelova putanja u hijerarhijskom procesu pretraživanja, ali zahtijeva dodatnu memoriju za predodređene parcijalne putanje. Međutim, materijalizacija cijena između svih parova čvorova može biti memorijski prezahtjevna [1], ako se radi o velikom broju čvorova u grafu [50]. Mnogo stroži zahtjev na vrijeme izračuna algoritma jest pri procesu dinamičkog planiranja. Problem većine algoritama koji imaju sposobnost izvršavanja u stvarnom vremenu jest neoptimalnost izračunate putanje. D. Cagigas u [18] koristi algoritam D* i pretvara ga u hijerarhijski algoritam D* (HD*), koji koristi skup optimalnih parcijalnih putanja između ključnih točaka H-grafa. Algoritam HD* unapređuje učinkovitost algoritma D* u velikim prostorima, ali optimalnost putanje ovisi o kreiranju H-grafa. Iako algoritam HD* koristi skup optimalnih parcijalnih putanja, nije osigurano da je globalna putanja sastavljena kao unija optimalnih parcijalnih putanja također optimalna. Dodatno, algoritam HD* koristi heuristiku prema cilju. Međutim, heuristika prema startu mogla bi ubrzati proces dinamičkog planiranja [127].

U ovome je radu na osnovi D. Cagigasova algoritma HD*, izveden novi hijerarhijski algoritam planiranja putanje nazvan fokusiranim hijerarhijskim D* (FHD*). Za razliku od izvornog algoritma HD*, algoritam FHD* jamči optimalnost globalne putanje i zahtijeva znatno manje vremena za dinamičko planiranje putanje. To je postignuto s nekoliko modifikacija izvornog algoritma HD*: (i) optimalnim postavljanjem tzv. mosnih čvorova potrebnih za stvaranje hijerarhije, (ii) organizacijom hijerarhije koja omogućuje optimalno horizontalno i vertikalno planiranje putanje i (iii) usmjeravanjem dinamičkog pretraživanja oko optimalne putanje, što smanjuje pretraženo područje bez gubitka optimalnosti. algoritam FHD* testiran je u unutarnjem prostoru s mnogo prostorija i uspoređen s izvornim algoritmom HD* te nehijerarhijskim algoritmima D* i FD* pod istim uvjetima. Algoritam FHD* znatno nadmašuje ostale algoritme s obzirom na vrijeme izračuna. Dodatno, može se jednostavno proširiti za planiranje putanje između različitih katova ili zgrada.

Struktura ovog poglavlja je sljedeća. Odlomak 3.1 opisuje koncept i definicije H-Grafa prema [19] i [18]. Odlomak 3.2 opisuje izvorni algoritam HD*. Odlomak 3.3 predstavlja novi algoritam FHD*, kao poboljšanje algoritma HD*. Odlomak 3.4 predstavlja metodu automatske izgradnje hijerarhijske karte. U odlomku 3.5 dani su eksperimentalni rezultati algoritma FHD* uspoređeni s rezultatima dobivenim algoritmima HD*, D* i FD* pod istim uvjetima.

3.1 Hijerarhijska karta

Ovdje iznosimo definiciju H-grafa prema [19] i [18] s našim notacijama i nekim dodatnim objašnjenjima.

3.1.1 H-graf

H-graf je skup razina hijerarhijske apstrakcije $\mathcal{R} = \{R_0, R_1, ..., R_D\}$, gdje je D dimenzija hijerarhije. R_0 je "korijenska" razina, koja predstavlja najapstraktniji opis prostora, tj. najvišu razinu hijerarhijske apstrakcije. Razina R_D predstavlja najdetaljniji opis prostora, tj. najnižu razinu hijerarhijske apstrakcije. Na svakoj razini $R_i \in \mathcal{R}$ postoji graf $\mathcal{G}_i(\mathcal{N}_i, \mathcal{A}_i, \mathcal{C}_i, \mathcal{W}_i, \mathcal{T}_i)$, gdje je $\mathcal{N}_i$ skup čvorova, $\mathcal{A}_i$ skup lukova, $\mathcal{C}_i$ skup Kartezijevih koordinata za $\mathcal{N}_i$, $\mathcal{W}_i$ skup težina za $\mathcal{A}_i$ i $\mathcal{T}_i$ skup parcijalnih putanja pridruženih $\mathcal{N}_i$. Indeks i označava razinu R_i. Unija grafova $\mathcal{G}_0, \mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_D$ je graf $\mathcal{G}(\mathcal{N}, \mathcal{A}, \mathcal{C}, \mathcal{W}, \mathcal{T})$, gdje je $\mathcal{N} = \mathcal{N}_0 \cup \mathcal{N}_1 \cup \cdots \cup \mathcal{N}_D$, $\mathcal{A} = \mathcal{A}_0 \cup \mathcal{A}_1 \cup \cdots \cup \mathcal{A}_D$, $\mathcal{C} = \mathcal{C}_0 \cup \mathcal{C}_1 \cup \cdots \cup \mathcal{C}_D$, $\mathcal{W} = \mathcal{W}_0 \cup \mathcal{W}_1 \cup \cdots \cup \mathcal{W}_D$, $\mathcal{T} = \mathcal{T}_0 \cup \mathcal{T}_1 \cup \cdots \cup \mathcal{T}_D$.

Luk $a(n_J, n_K, w_H) \in \mathcal{A}$ definiran je pomoću tri elementa n_J, n_K, w_H, gdje su $n_J, n_K \in \mathcal{N}$, $n_J \neq n_K$ i $w_H \in \mathcal{W}$. Lukovi su neusmjereni i dopuštaju navigaciju u oba smjera između dvaju čvorova. Definirani su samo između nekih čvorova. Za razliku od ostalih modela H-grafa, primjerice [33, 34], lukovi ne sadrže druge lukove u dubljoj razini hijerarhije. No, postoje lukovi kojima su čvorovi n_J, n_K iz različitih razina. Tada taj luk pripada razini čvora iz niže razine. Primjerice, ako je $n_J \in \mathcal{N}_j$ i $n_K \in \mathcal{N}_k$, $0 < j < k \leq D$, tada je $a(n_J, n_K, w_H) \in \mathcal{A}_k$.

Kartezijeve koordinate $c \in \mathcal{C}$ definirane su uređenim parom (x, y), gdje su $x, y \in \mathbb{R}$. Kartezijeve su koordinate pridružene samo nekim čvorovima iz $\mathcal{N}$ i koriste se u heurističnoj funkciji za algoritam planiranja putanje.

Težina $w \in \mathcal{W}$ je realni broj ($w_I \in \mathbb{R}$). Težine su pridružene svim postojećim lukovima. Predstavljaju cijenu putovanja po luku i koriste su u funkciji cijene putanje u algoritmu planiranja putanje.

3.1.2 Kategorije čvorova i njihova svojstva

Čvorovi su podijeljeni u četiri skupine: *potkartni* čvorovi, *krajnji* čvorovi, čvorovi *raskrižja* i *mosni* čvorovi.

Potkartni čvorovi su disjunktni podskupovi čvorova iz niže razine hijerarhijske apstrakcije. Skup potkartnih čvorova označavamo sa $\mathcal{K} \subset \mathcal{N}$. Kažemo da je čvor $k \in N_i$ potkartni čvor iz razine R_i, $(0 \leq i < D)$, ako je sastavljen od $K > 1$ čvorova iz niže razine R_j, gdje je $j = i+1$, odnosno $k = \{n_1, n_2, \ldots, n_K\} \subset N_j$. *Potkartni čvor roditelj* čvoru $n \in \mathcal{N}_i$, $0 < i \leq D$, je onaj čvor $k \in \mathcal{N}_{i-1} \cap \mathcal{K}$ takav da je $n \in k$. Definirajmo funkciju $roditelj : \mathcal{N} \to \mathcal{K}$ koja određuje potkartni čvor roditelj čvoru n

$$roditelj(n) := k \mid n \in k. \tag{3.1}$$

Definirajmo funkciju $razina : \mathcal{N} \to \{0, \ldots, D\}$ koja određuje u kojoj se razini nalazi čvor n

$$razina(n) := i \mid n \in \mathcal{N}_i, \tag{3.2}$$

uz napomenu da se čvor $n \in \mathcal{N}$ može nalaziti samo u jednoj razini. Primijetimo da vrijedi sljedeće:

$$\text{ako je } roditelj(n) = k \text{ tada je } razina(n) = razina(k) + 1. \tag{3.3}$$

Potkartni se čvorovi nalaze u svim razinama hijerarhijske apstrakcije osim u najnižoj razini R_D. Svi čvorovi iz iste razine raspoređeni su u potkartne čvorove na višoj razini, odnosno vrijedi sljedeće:

$$\text{za } \forall i \in \{0, \ldots, D-1\} \text{ i za } \forall n \in \mathcal{N}_{i+1}\ \exists_1 k \in \mathcal{N}_i \text{ tako da } n \in k \subset \mathcal{N}_{i+1}. \tag{3.4}$$

Dakle, svaki čvor n ima samo jedan potkartni čvor roditelj k, budući da su potkartni čvorovi disjunktni skupovi.

Krajnji čvorovi su svi čvorovi koji se mogu odabrati kao start i cilj za algoritam planiranja putanje.

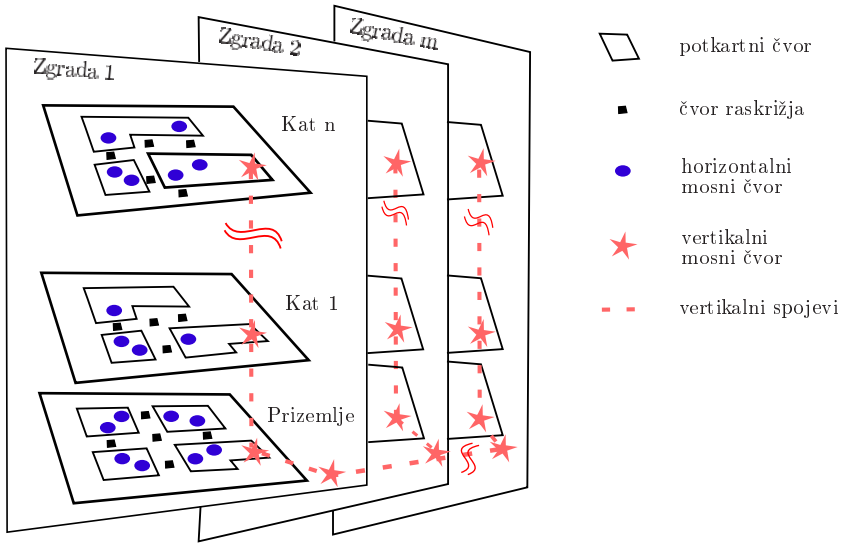
Čvorovi *raskrižja* su podciljevi koji predstavljaju križanja putanja i skretnice na putanjama.

Kažemo da je $m \in \mathcal{N}_j$ *mosni* čvor iz razine R_j, $(1 < j \leq D)$, ako postoji čvor $n \in \mathcal{N}_i$ iz razine R_i, $i = j - 1$, i luk $a(m, n, w) \in \mathcal{A}$. Mosni se čvorovi nalaze u svim razinama hijerarhijske apstrakcije osim u prve dvije razine R_0 i R_1. Pretpostavka je da se na razini R_0 nalazi samo jedan potkartni čvor. Svaki mosni čvor $m \in \mathcal{N}_j$, $(1 < j \leq D)$, ima svoj potkartni čvor roditelj $k \in \mathcal{N}_i$, $i = j - 1$, koji se zajedno sa čvorom $n \in \mathcal{N}_i$ takvim da $\exists a(m, n, w) \in A$ nalazi u potkartnom čvoru $h \in \mathcal{N}_{i-1}$, odnosno $m \in k$ i $n, k \in h$. Prema tome, možemo reći da mosni čvorovi povezuju potkartni čvor s njegovim potkartnim čvorom roditeljem. Skup mosnih čvorova označavamo sa $\mathcal{M} \subset \mathcal{N}$. Skup svih mosnih čvorova unutar potkartnog čvora k, označen $\mathcal{M}_k \subseteq \mathcal{M}$, određuje se prema:

$$\mathcal{M}_k = \mathcal{M} \cap k. \tag{3.5}$$

Postoje dvije vrste mosnih čvorova: *horizontalni mosni čvorovi* i *vertikalni mosni*

čvorovi. Horizontalni mosni čvorovi prate prethodnu definiciju, a vertikalni se mosni čvorovi razlikuju od horizontalnih mosnih čvorova po tome što spajaju dva potkartna čvora koji predstavljaju dva kata u zgradi. Ulazi u dizala modelirani su kao vertikalni mosni čvorovi. Ti čvorovi omogućuju planiranje putanje između različitih katova u zgradi ili čak između različitih katova koji pripadaju različitim zgradama. Primjer modeliranja više zgrada H-grafom u kojem su naznačeni horizontalni i vertikalni mosni čvorovi prikazan je na slici 3.1. Katovi su prikazani kao potkartni čvorovi unutar kojih su naznačeni ostali čvorovi u nižoj razini hijerarhijske apstrakcije. Lukovi među čvorovima nisu naznačeni, osim spojeva vertikalnih mosnih čvorova crtkanom linijom.



Slika 3.1. *Primjer H-grafa više zgrada.*

3.1.3 Parcijalne putanje

Neka je parcijalna putanja $\mathcal{T} = \mathcal{T}(a, b, k)$ definirana pomoću mosnih čvorova $a, b \in \mathcal{M}$ i potkartnog čvora $k \in \mathcal{K}$ takva da vrijedi:

$$\begin{aligned} \mathcal{T}[1] = a, \quad \mathcal{T}[|\mathcal{T}|] = b, \\ \mathcal{T}[i] \in k, \quad i = 1, \ldots, |\mathcal{T}|, \\ a(\mathcal{T}[j], \mathcal{T}[j+1], w_{(j,j+1)}) \in \mathcal{A}, \quad j = 1, \ldots, |\mathcal{T}| - 1. \end{aligned} \tag{3.6}$$

Duljina parcijalne putanje definirana je kao $|\mathcal{T}|$. Cijena parcijalne putanje $\mathcal{T}$ definirana je kao zbroj težina lukova na putanji

$$c(\mathcal{T}) := \sum_{i=1}^{|\mathcal{T}|-1} w_{(i,i+1)}. \tag{3.7}$$

Među svim putanjama između čvorova a i b, takvima da su svi elementi putanje unutar potkartnog čvora k, parcijalna putanja $\mathcal{T}(a,b,k)$ ima najmanju cijenu putanje $c(\mathcal{T})$. Svaki potkartni čvor $k \in \mathcal{N}_i$, $(0 \leq i < D)$ ima pridružen skup predodređenih parcijalnih putanja $\mathcal{T}_k \in \mathcal{T}_i$, $(0 \leq i \leq D)$. Računaju se tri tipa parcijalnih putanja:

Tip 1) Putanja koja spaja dva mosna čvora unutar potkartnog čvora k,

$$\mathcal{T}(a,b,k),\ a,b \in \mathcal{M} \cap k. \tag{3.8}$$

Tip 2) Putanja koja spaja mosni čvor iz potkartnog čvora k s mosnim čvorom potkartnog čvora roditelja $roditelj(k)$,

$$\mathcal{T}(a,b,k),\ a \in \mathcal{M} \cap k,\ b \in \mathcal{M} \cap\ roditelj(k). \tag{3.9}$$

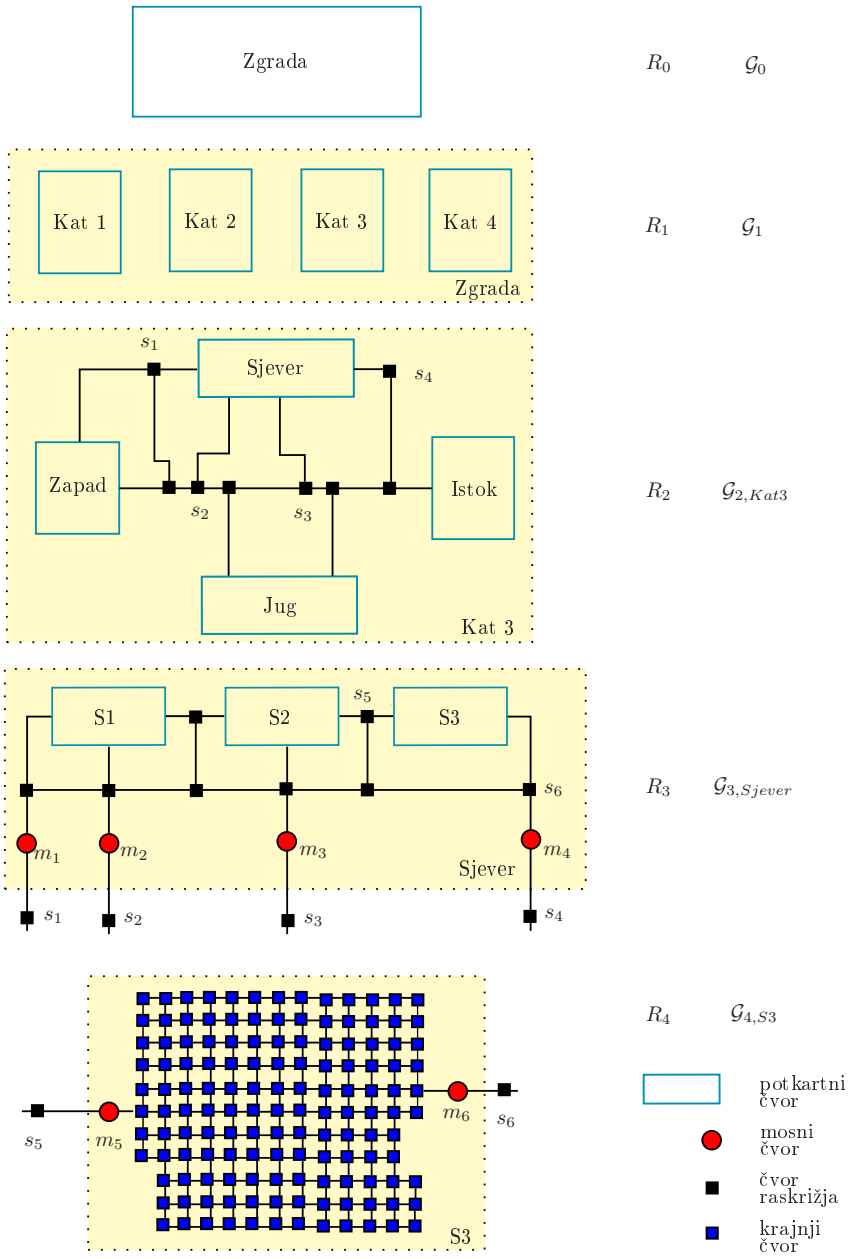
Tip 3) Putanja koja spaja dva mosna čvora iz "bratskih" potkartnih čvorova sadržana u h. Potkartni čvorovi k_1 i k_2 nazivaju se "bratskim" čvorovima ako imaju istog roditelja h, odnosno $roditelj(k_1) = roditelj(k_2) = h$.

$$\mathcal{T}(a,b,h),\ a \in \mathcal{M} \cap k_1,\ b \in \mathcal{M} \cap k_2,\ k_1,k_2 \in h. \tag{3.10}$$

Uz ovako određene parcijalne putanje (tzv. materijalizacija cijena) izbjegnuto je postupno korigiranje putanje u dubljim razinama hijerarhije. Negativno je obilježje materijalizacije cijena dodatno memorijsko spremište parcijalnih putanja, koje je nadoknađeno znatno manjim računskim vremenom potrebnim za hijerarhijsko planiranje putanje. Dodatno, H-graf ima svojstvo jednostavnog proširenja hijerarhije. Ako se proširi karta zgrade, primjerice dodavanjem katova ili čak cijele nove zgrade postojećoj karti, H-graf se jednostavno proširuje dodavanjem novih čvorova i parcijalnih putanja ili čak novih razina.

3.1.4 Modeliranje prostora H-grafom

Prema primjeru iz [19] na slici 3.2 prikazana je hijerarhijska karta jedne zgrade. Na najapstraktnijoj razini (R_0) nalazi se graf $\mathcal{G}_0$ koji se sastoji od jednog potkartnog čvora – *Zgrada*. Na nižoj razini, R_1, nalazi se graf $\mathcal{G}_1$ sadržan od potkartnih čvorova koji predstavljaju katove u zgradi – *Kat 1*, *Kat 2*, *Kat 3* i *Kat 4*. Na razini R_2 nalazi se graf $\mathcal{G}_2$ koji je unija grafova svakog od katova. Na slici je prikazana unutrašnjost trećeg kata – graf $\mathcal{G}_{2,Kat3} \subset \mathcal{G}_2$. Kat je podijeljen na dijelove. Svaki dio kata predstavljen je potkartnim čvorom. Njihovi nazivi odgovaraju stranama svijeta: *Istok*, *Zapad*, *Sjever* i *Jug*. Na ovoj su razini prisutni i čvorovi raskrižja. S označenim čvorovima raskrižja povezani su mosni čvorovi iz niže razine. Na razini R_3 nalaze se detalji svakog dijela kata, za svaki od katova. Na slici je prikazan graf $\mathcal{G}_{3,Sjever} \subset \mathcal{G}_3$ koji predstavlja detalje sjevernog dijela trećeg kata. Potkartni čvorovi u njemu predstavljaju sobe *S1*, *S2* i *S3*. U ovom su grafu prisutni mosni čvorovi, koji se vežu za čvorove raskrižja iz grafa $\mathcal{G}_{2,Kat3}$.



Slika 3.2. *Primjer H-grafa zgrade s četiri kata.*

Radi jasnoće, njihova je veza prikazana u razini R_3 izvan pravokutnika koji omeđuje graf $\mathcal{G}_{3,Sjever}$. Razina R_4 predstavlja najdetaljniju razinu na kojoj se nalazi graf $\mathcal{G}_4$. Na toj se razini nalaze detalji potkartnih čvorova razine R_3 i nema novih potkartnih čvorova. Prikazana je struktura sobe S3, graf $\mathcal{G}_{4,S3} \subset \mathcal{G}_4$. Tu se nalaze krajnji čvorovi i dva mosna čvora koji su vezani za čvorove raskrižja iz grafa $\mathcal{G}_{3,Sjever}$.

U primjeru su u razini R_4 prikazane detaljno samo unutrašnjosti soba, dok su hodnici prikazani na višim razinama kao niz čvorova raskrižja koji povezuju sobe (razina R_3),

odnosno dijelove kata (razina R_2). Svi prikazani mosni čvorovi su horizontalni mosni čvorovi. Vertikalni mosni čvorovi mogu se nalaziti na svim razinama osim na R_0 i R_1 (vrijedi za sve mosne čvorove). Primjerice, vertikalni mosni čvor može se nalaziti u razini R_2 nadovezan na jedan čvor raskrižja. Tada bi povezivao svoj potkartni čvor Kat 3 s potkartnim čvorom roditeljem Zgrada. Isto tako, vertikalni mosni čvorovi nalazili bi se na istoj razini i u ostalim potkartnim čvorovima – katovima. Na razini R_1 bi se trebao nalaziti jedan čvor raskrižja na koji bi se vezali svi vertikalni mosni čvorovi iz svih katova.

Postavljanje vertikalnog mosnog čvora određeno je samom strukturom zgrade i nije uvijek moguće postaviti vertikalni mosni čvor u odgovarajuću razinu. Primjerice, da bi se vertikalni mosni čvor mogao postaviti u razinu R_2 mora se pozicija dizala nalaziti u dijelu prostora koji još nije opisan u nižim razinama hijerarhijske apstrakcije, odnosno, morao bi se nalaziti u neopisanom dijelu hodnika. Međutim, ako se dizala nalaze unutar jednog dijela kata, primjerice zapadnog dijela, tada ne mogu povezivati druge katove kao vertikalni mosni čvorovi jer se vežu za druge čvorove na višoj razini koji su unutar istog kata. Čvor u razini iznad s kojim bi se vezao vertikalni mosni čvor morao bi biti na drugom katu, odnosno u potkartnom čvoru koji obuhvaća sve katove. Drugim riječima, vertikalni se čvorovi moraju nalaziti u jednoj razini ispod definiranih potkartnih čvorova katova.

Umjesto da značajno mijenjaju strukturu hijerarhije, u [19, 18] su vertikalne mosne čvorove tretirali drugačije od horizontalnih mosnih čvorova. Prilagodili su hijerarhijsko pretraživanje tako što su vertikalne mosne čvorove proglasili ciljevima i startevima. Na taj su način razložili problem planiranja putanje između dva kata x i y na dva potproblema: 1) planiranje putanje od starta do vertikalnog mosnog čvora na katu x i 2) planiranje putanje od vertikalnog mosnog čvora na katu y do cilja. Dizala su modelirali kao niz vertikalnih mosnih čvorova od kojih se svaki nalazi na ulazu u dizalo. Planiranje putanje između dviju zgrada još je složeniji. Neka je start u zgradi A na katu x, a cilj u zgradi B na katu y. Problem planiranja putanje razlaže se na tri potproblema: 1) planiranje putanje od starta do vertikalnog mosnog čvora u zgradi A na katu n, 2) planiranje putanje od vertikalnog mosnog čvora u zgradi A u prizemlju do vertikalnog mosnog čvora u zgradi B u prizemlju i 3) planiranje putanje od vertikalnog mosnog čvora u zgradi B na katu y do cilja. U slučaju da u zgradama postoji više dizala koji nisu blizu jedan drugom, problem planiranja putanje postaje još složeniji. U [18] taj računski zahtjevan problem rješavaju odabirom jednog (bilo kojeg) vertikalnog mosnog čvora, iako je tako narušena optimalnost putanje.

3.2 Hijerarhijski algoritam D*

Ključni dio Cagigasovog algoritma HD* je skup predodređenih parcijalnih putanja (materijalizacija cijena). Za svaki potkartni čvor određena su tri tipa parcijalnih putanja (3.8)-(3.10) algoritmom A*.

3.2.1 Inicijalno planiranje putanje

Izračun inicijalne putanje opisan je u [19]. Algoritam nastoji naći najbolji skup putanja *kostura* između potkartnih čvorova u kojima se nalaze start i cilj, gdje je putanja kostur takva putanja koja se sastoji samo od mosnih čvorova (označena s Π). Označimo razinu tzv. zajedničkog potkartnog čvora s R_i. Zajedničkim potkartnim čvorom zovemo onaj čvor koji obuhvaća start i cilj u nekim nižim razinama hijerarhijske apstrakcije. U najgorem slučaju, zajednički potkartni čvor je korijenski čvor.

Algoritam 3.1: dodaj-procijenjene-putanje$(S, \mathcal{M}_k)$

```
za ∀n ∈ M_k
    P_i ← [S, n]
    c(P_i) ← heuristika(S, n)
    Π ← Π ∪ P_i
kraj
vrati Π
```

Algoritam 3.2: odredi-skup-putanja-kostura(S, r)

```
k ← roditelj(S)
M_k ← M ∩ k
Π ← dodaj-procijenjene-putanje(S, M_k)
dok razina(k) > r  //razina zajedničkog potkartnog čvora
    Π_n ← ∅
    za ∀n ∈ M ∩ roditelj(k)
        P_i ← ∅
        c(P_i) ← ∞
        za ∀P_j ∈ Π
            m ← P_j[|P_j|]
            P_x ← P_j ∪ T(m, n, k)  //Parcijalna putanja tip 2
            c(P_x) ← c(P_j) + c(T(m, n, k))
            ako c(P_x) < c(P_j)
                P_i ← P_x
            kraj
        kraj
        Π_n ← Π_n ∪ P_i
    kraj
    Π ← Π_n
    k ← roditelj(k)
kraj
vrati Π
```

Skup putanja kostura stvara se tako da se minimizira akumulirana cijena ili procjena cijene između svaka dva pretražena čvora. Prvo se stvore putanje kosturi od starta (cilja) do mosnih čvorova iz potkartnog čvora roditelja od starta, odnosno cilja (algoritam 3.1). Te putanje nisu parcijalne putanje pa se zato koristi procjena cijene putanja.

Dalje se putanje kosturi nadograđuju u višoj hijerarhijskoj razini tako što se korištenjem parcijalnih putanja tipa 2 (3.9) dodaju najbolje parcijalne putanje koje spajaju potkartni čvor roditelj od starta (cilja) s njegovim roditeljem (algoritam 3.2). Radi se u dvije razine: u razini od "djeteta" i u razini od "roditelja". Promatraju se sve moguće parcijalne putanje tipa 2 između tih dviju razina i odabiru se one koje minimiziraju akumuliranu cijenu putanja kostura. Proces se rekurzivno ponavlja kroz nekoliko razina hijerarhijske apstrakcije sve dok se ne dosegne razina R_i.

Sljedeći je korak povezivanje obaju skupova putanja kostura (algoritam 3.3). Skupovi se povezuju nedostajućim parcijalnim putanjama – tip 2 ili tip 3.

Algoritam 3.3: povezivanje-skupova-putanja-kostura(Π_S, Π_G)

za $\forall \mathcal{P}_S \in \Pi_S$
  za $\forall \mathcal{P}_G \in \Pi_G$
    $n \leftarrow \mathcal{P}_S[|\mathcal{P}_S|]$
    $m \leftarrow \mathcal{P}_G[|\mathcal{P}_G|]$
    $k \leftarrow roditelj(n)$
    $\mathcal{T} \leftarrow \mathcal{T}(n, m, k)$
    ako $\mathcal{T} \neq \emptyset$ *//parcijalna putanja tip 2*
      $\mathcal{P} \leftarrow \mathcal{P}_S \cup \mathcal{T} \cup \mathcal{P}_G$
      $c(\mathcal{P}) \leftarrow c(\mathcal{P}_S) + c(\mathcal{T}) + c(\mathcal{P}_G)$
    inače
      $k \leftarrow roditelj(k)$
      $\mathcal{T} \leftarrow \mathcal{T}(n, m, k)$
      ako $\mathcal{T} \neq \emptyset$ *//parcijalna putanja tip 3*
        $\mathcal{P} \leftarrow \mathcal{P}_S \cup \mathcal{T} \cup \mathcal{P}_G$
        $c(\mathcal{P}) \leftarrow c(\mathcal{P}_S) + c(\mathcal{T}) + c(\mathcal{P}_G)$
      inače
        $\mathcal{P} \leftarrow \mathcal{P}_S \cup \mathcal{P}_G$
        $c(\mathcal{P}) \leftarrow c(\mathcal{P}_S) + c(\mathcal{P}_G) + heuristika(n, m)$
      kraj
    kraj
    $\Pi \leftarrow \Pi \cup \mathcal{P}$
  kraj
kraj
vrati Π

Na kraju se cijeli skup putanja kostura nadomjesti pravim putanjama (izračunaju se putanje između točaka na putanji kosturu) i odabere najbolja putanja. Ovo je tipičan primjer procesa odozdo-gore koji se koristi u hijerarhijskim grafovima. Algoritam rješava i slučaj da se start i cilj ne nalaze u istim razinama. Algoritam se ponaša slično A* algoritmu jer radi s procjenama cijena dijelova putanje. Od starta do mosnog čvora je jedna procjena (heuristika), a dalje se do zajedničke razine radi s materijaliziranim cijenama. Isto se ponavlja i s ciljem. Dakle nastoji se minimizirati ukupna cijena $f = c + heuristika$, gdje je c akumulirana cijena putanje.

3.2.2 Dinamičko planiranje putanje

Dinamičko planiranje putanje započinje kada se dogodi promjena u prostoru. Dijelovi prostora koji su postali zauzeti nestaju iz grafa (budući da čvorovi u grafu predstavljaju samo slobodan dio prostora mobilnog robota). Nestankom čvora iz grafa "prekidaju" se lukovi koji su definirani tim čvorom.

Koristi se skup Π kao spremište privremenih rješenja - novih dijelova putanja koje vode do cilja. U skup Π početno se dodaje startni čvor (koji predstavlja trenutnu poziciju robota) i svi čvorovi koji imaju definiran luk s njim. Ti čvorovi zajedno sa startnim čvorom čine početne dijelove novih putanja. Dodaju se tako da minimiziraju cijenu $f = c + g$, gdje je c cijena putanje iz skupa Π, a g procjena ostatka putanje do cilja (pretraživanje algoritmom A*).

Algoritam 3.4: hijerarhijsko-D*-pretraživanje($\mathcal{P}, \mathcal{P}_I, \Pi$)

$o \leftarrow \mathcal{P}[|\mathcal{P}|]$ *//Zadnji čvor na najboljoj putanji je mosni čvor*
$i \leftarrow 1$
$n \leftarrow \mathcal{P}_I[1]$
dok $n \notin \mathcal{M}$ ili $roditelj(n) \neq roditelj(o)$
    $i \leftarrow i + 1$
    $n \leftarrow \mathcal{P}_I[i]$
kraj *//n je mosni čvor i "brat" od čvora o*
ponavljaj
    $i \leftarrow i + 1$
    $n \leftarrow \mathcal{P}_I[i]$
dok $n \notin \mathcal{M}$ *//n je sljedeći mosni čvor koji nije "brat" od čvora o*
$k \leftarrow roditelj(n)$ *//k je sljedeći potkartni čvor koji se mora dosećи*
$h \leftarrow roditelj(o)$
ako $roditelj(h) = k$
    $p \leftarrow h$ *//Parcijalna putanja tip 2 podiže putanju u višu razinu*
inače ako $roditelj(k) = h$
    $p \leftarrow k$ *//Parcijalna putanja tip 2 spušta putanju u nižu razinu*
inače *//*$roditelj(k) = roditelj(h)$
    $p \leftarrow roditelj(k)$ *//Parcijalna putanja tip 3, ista razina*
kraj
za $\forall m \in \mathcal{M} \cap k$
    $\mathcal{T} \leftarrow \mathcal{T}(o, m, p)$
    $\mathcal{P}_n \leftarrow \mathcal{P} \cup \mathcal{T}$ *//Najbolja putanja* $\mathcal{P}$
    $c(\mathcal{P}_n) \leftarrow c(\mathcal{P}) + c(\mathcal{T})$
    $\Pi \leftarrow \Pi \cup \mathcal{P}_n$
kraj
vrati Π

Najbolja putanja $\mathcal{P}$ (s obzirom na cijenu $f = c+h$) oduzima se iz skupa Π. Dalje se razmatra zadnji čvor najbolje putanje $\mathcal{P}$, označen s o. Postoje tri slučaja koje algoritam rješava potprocedurama: 1) ako je o mosni čvor, izvodi se algoritam 3.4; 2) ako je o potkartni čvor, izvodi se algoritam 3.5; 3) inače, nastavlja se pretraživanje algoritmom A* od čvora o. Taj se postupak ponavlja, dakle oduzimanje najbolje putanje iz skupa

Π i izvršavanje jedne od triju procedura, sve dok se ne pronađe čvor koji se nalazi na inicijalnoj putanji $\mathcal{P}_I$ ili dok se ne dohvati čvor iz ciljnog potkartnog čvora. U zadnjem se slučaju na najbolju putanju nadoveže parcijalna putanja do cilja ako postoji ili se čvor po čvor pretraži algoritmom A* (procedura 3).

Algoritam 3.5: potkartno-D*-pretraživanje($\mathcal{P}, \Pi$)

1: $o \leftarrow \mathcal{P}[|\mathcal{P}|]$ *//Zadnji čvor na najboljoj putanji je potkartni čvor*
2: $r \leftarrow \mathcal{P}[|\mathcal{P}| - 1]$ *//Predzadnji čvor na najboljoj putanji je čvor raskrižja*
3: $m \leftarrow m \in \mathcal{M} \cap o \mid \exists a(m, r, w_{mr}) \in \mathcal{A}$
4: za $\forall n \in \mathcal{M} \cap o,\ n \neq m$
5: $\mathcal{T} \leftarrow \mathcal{T}(m, n, o)$ *//Parcijalna putanja tip 1*
6: $p \leftarrow p \in \mathcal{N} \mid \exists a(n, p, w_{np}) \in \mathcal{A},\ roditelj(p) = roditelj(o)$
7: $\mathcal{P}_n \leftarrow \mathcal{P} \setminus o \cup \mathcal{T} \cup p$ *//Čvor o zamjenjuje s parcijalnom putanjom*
8: $c(\mathcal{P}_n) \leftarrow c(\mathcal{P}) + c(\mathcal{T}) + w_{np}$
9: $\Pi \leftarrow \Pi \cup \mathcal{P}_n$
10: kraj
11: vrati Π

U proceduri 1) nastoji se povezati potkartni čvor roditelj čvora o s potkartnim čvorom na inicijalnoj putanji $\mathcal{P}_I$. Povezivanje se radi na način da se u inicijalnoj putanji nađe prvi mosni čvor koji se ne nalazi u istom potkartnom čvoru kao i o. Tada se potkartni čvorovi od oba mosna čvora povežu svim mogućim parcijalnim putanjama između njihovih mosnih čvorova - to su tip 2 ili tip 3, ovisno o tome u kojoj je razini sljedeći mosni čvor na inicijalnoj putanji. Valja napomenuti da se inicijalna putanja sastoji od čvorova koji se nalaze na različitim razinama hijerarhijske apstrakcije. Razine rastu od najniže u kojoj je start do razine zajedničkog potkartnog čvora i zatim se spuštaju do najniže razine u kojoj je cilj. Sve moguće parcijalne putanje nadograđuju se na najbolju putanju $\mathcal{P}$ i dodaju skupu Π.

Za vrijeme osnovnog pretraživanja algoritmom A* (procedura 3) moguće je naići na potkartni čvor. Primjera radi, neka je mosni čvor m spojen sa čvorom raskrižja r u višoj razini u kojoj se spoj vidi kao luk između čvora raskrižja r i potkartnog čvora roditelja o od mosnog čvora m. U proceduri 2) čvor o, koji je potkartni čvor, "rastvara" se i iz njega se izdvaja skup onih parcijalnih putanja, koje sadrže mosni čvor m koji je spojen sa čvorom raskrižja r. Svakoj parcijalnoj putanji dodaju se i čvorovi raskrižja s kojima su vezani na višoj razini. Takve putanje nadovezane na najbolju putanju $\mathcal{P}$ dodaju se skupu Π. Zadnji čvorovi ovih putanja opet su u istoj razini hijerarhijske apstrakcije kao i potkartni čvor o. Time se pretraživanje ponovo nastavlja u prethodnoj razini hijerarhijske apstrakcije. Dakle, u ovoj se proceduri pretraživanje samo privremeno spušta u nižu razinu zbog supstitucije podkratnog čvora njegovim skupom parcijalnih putanja.

3.2.3 Diskusija

Zbog više razina hijerarhijskih apstrakcija algoritam brže izračunava putanju od algoritama pretraživanja jedno-razinskog grafa. Međutim, pogoršana je kvaliteta putanja, odnosno, putanje nisu optimalne. Hijerarhijski algoritam pretraživanja prema [19, 18] ne materijalizira sve moguće putanje (inače bi algoritam određivao optimalnu putanju) nego štedi na memorijskom i vremenskom izračunu. Međutim, nisu potrebne sve materijalizacije cijena. Algoritam bi mogao određivati optimalnu putanju kada bi mosni čvorovi pokrivali sve mogućnosti prolaska putanje iz jednog potkartnog čvora u drugi.

U [19] iznose da je veličina potkartnog čvora, odnosno broj čvorova u potkartnom čvoru, strogo vezana s optimalnošću i računskom učinkovitošću. Mali potkartni čvorovi (mali skupovi) impliciraju nisku računalnu potrošnju, ali kvaliteta rješenja opada i obrnuto. Također iznose da je najbolje prve potkartne čvorove odrediti tako da opisuju sobe i da sadrže mali broj čvorova.

U slučaju malog broja čvorova u grafu, algoritam je lošiji od nehijerarhijskog algoritma A*, jer izvodi više operacija zbog složenosti H-grafa (rekurzivno vraćanje iz više u nižu razinu zbog dohvaćanja parcijalnih putanja i rastvaranja potkartnih čvorova).

Pozicija mosnih čvorova nije razmatrana, iako je ona ključna za efikasnu izvedbu hijerarhijskog planiranja. Ako se mosni čvorovi postave na proizvoljna mjesta u prostoru, mogu se naći izvan optimalne putanje između starta i cilja. U tom slučaju unija parcijalnih optimalnih putanja između mosnih čvorova ne daje optimalnu putanju od starta do cilja. Dodatno, u najdetaljnijoj razini ne pokrivaju cijeli prostor mobilnog robota, već samo sobe. Za gibanje mobilnog robota nužno je da slijedi najkraću putanju u svim dijelovima prostora i da svaka realna pozicija (diskretna) u prostoru odgovara jednom čvoru u najdetaljnijoj razini hijerarhijske apstrakcije. Prikladan izbor najdetaljnije razine bila bi mrežasta karta zauzeća, odnosno njezin graf.

Vertikalno dinamičko planiranje algoritmom HD* (između dva kata) zahtijeva dodatnu diskusiju. Razmotrimo sljedeći problem: potrebno je isplanirati putanju od starta koji se nalazi na jednom katu, do cilja koji se nalazi na drugom katu, a postoje dva međusobno udaljena dizala. Inicijalna putanja prolazi dizalom A, a alternativno je dizalo B. Ako se pojavi prepreka na putanji takva da je kraća putanja ona koja ide dizalom B dvije su mogućnosti: 1) odabrati ponovo dizalo A ili 2) odabrati alternativno dizalo B. Druga mogućnost znači za algoritam HD* potpuno preračunavanje putanje, jer su dizala pod-ciljevi i pod-startovi, odnosno, razbijen je problem planiranja putanje na dva dijela, od starta do dizala i od dizala do cilja. Dodatno, odabir ove mogućnosti ne jamči rješenje, jer možda ne postoji putanja do dizala B. Stoga se uvijek odabire dizalo s inicijalne putanje, a u slučaju da ne postoji rješenje, odabire se drugo dizalo. Time je narušena optimalnost vertikalnog planiranja putanje.

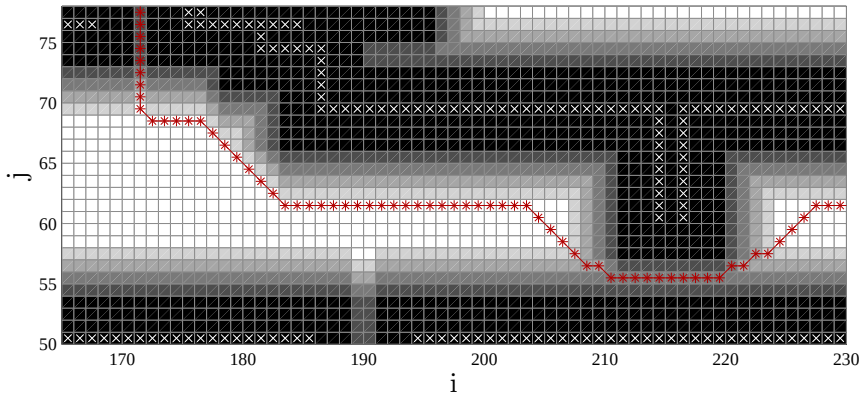
3.3 Fokusirani hijerarhijski algoritam D*

S ciljem osiguranja optimalnosti globalne putanje i smanjenja računske složenosti dinamičkog planiranja puta izvedene su tri bitne modifikacije izvornog algoritma HD*: (i) optimalno postavljanje mosnih čvorova, (ii) organizacija hijerarhije koja omogućuje optimalno horizontalno i vertikalno planiranje putanje i (iii) usmjeravanje dinamičkog pretraživanja oko optimalne putanje. Nazvali smo algoritam fokusiranim hijerarhijskim D* (FHD*).

3.3.1 Optimalno postavljanje mosnih čvorova

U [19] i [18] pozicije mosnih čvorova nisu razmatrane, iako one uvelike utječu na optimalnost izračunate putanje hijerarhijskim algoritmom planiranja putanje. Mosni čvorovi moraju se postaviti na određena mjesta u prostoru, tako da unija parcijalnih optimalnih putanja između mosnih čvorova daje optimalnu putanju. Stoga postavljamo mosne čvorove na mjesta između dva potkartna čvora kojima će prolaziti sve optimalne putanje od starta do cilja, gdje je start bilo koja točka u jednom potkartnom čvoru, a cilj bilo koja točka u drugom potkartnom čvoru.

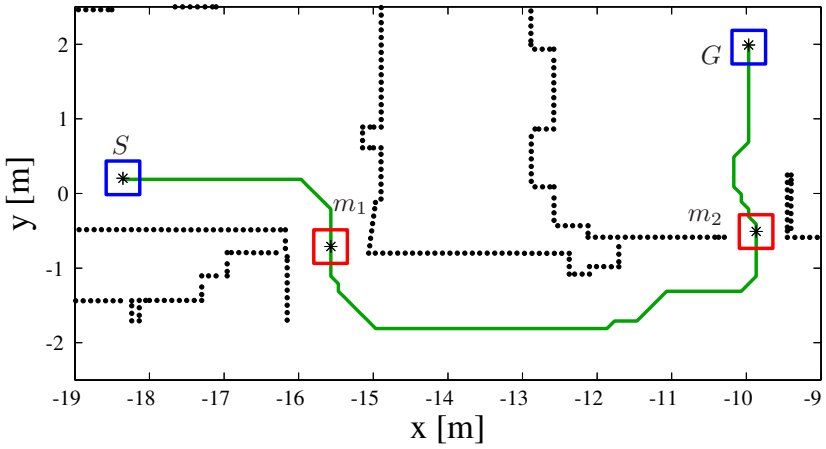
Pri slijeđenju putanje sigurnost mobilnog robota je od velikog značaja. U uskim je prolazima najsigurnija putanja ona koja prolazi kroz sredinu prolaza. Stoga kao najdetaljniju razinu R_D koristimo graf $\mathcal{G}_D$ konstruiran iz mrežaste karte zauzeća sa sigurnosnom maskom cijena (poglavlje 2.2.2). U uskim prolazima središnja polja imaju najniže vrijednosti sigurnosne cijene i stoga sve optimalne putanje, koji prolaze uskim prolazom, prolaze tim poljima, kao što je prikazano na slici 3.3.



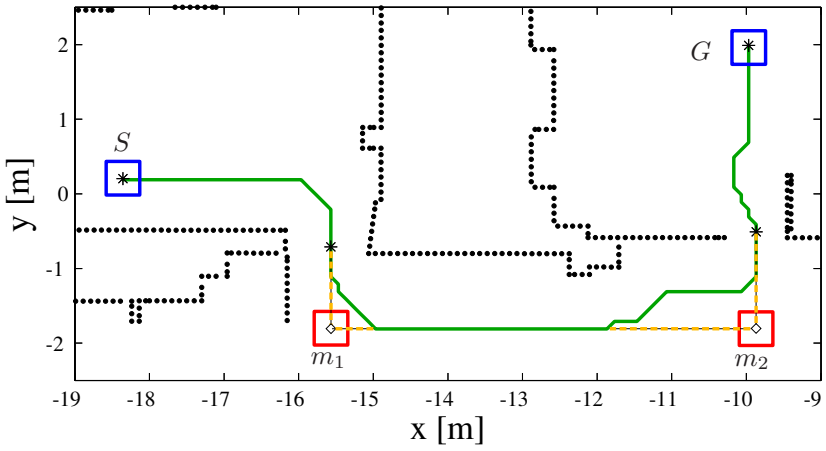
Slika 3.3. *Primjer optimalne putanje proračunate na grafu konstruiranom iz mrežaste karte zauzeća sa sigurnosnom maskom cijena.*

Na višoj razini R_{D-1} potkartni čvorovi predstavljaju sobe i hodnike. Prema tome, logično je postaviti mosne čvorove na sredinu prolaza kod vrata koja odvajaju sobe. Ako postoji paran broj polja u uskom prolazu, gledajući po širini uskog prolaza, onda su postavljena dva mosna čvora jedan do drugog tako da se njihov zajednički brid nalazi na

sredini uskog prolaza. Mosni čvorovi su na granici između dvije sobe i dijele ih obje sobe, odnosno, oni pripadaju i jednoj i drugoj sobi. Ako postoji putanja između dva potkartna čvora, optimalna putanja mora prolaziti kroz mosni čvor. Tada unija parcijalnih putanja između mosnih čvorova također daje optimalnu putanju. Optimalnost putanje leži u činjenici da jednom određena optimalna putanja sadrži parcijalne optimalne putanje prema Belmanovu načelu optimalnosti u kontekstu problema najkraće putanje [123]. Načelo kaže da ako postoji optimalna putanja $\mathcal{P}_1^*$ od čvora S do čvora G, koja sadrži čvor m, tada dio putanje $\mathcal{P}_1^*$ između m i G ujedno predstavlja optimalnu putanju $\mathcal{P}_2^*$ između m i G. Analogno i dio putanje $\mathcal{P}_1^*$ između S i m ujedno predstavlja optimalnu putanju $\mathcal{P}_3^*$ između S i m.



Slika 3.4. *Pozicija mosnih čvorova na optimalnim mjestima: unija parcijalnih putanja je optimalna putanja.*



Slika 3.5. *Loša pozicija mosnih čvorova: unija parcijalnih putanja ne daje optimalnu putanju.*

Na slici 3.4 prikazana je optimalna putanja određena algoritmom D* između starta

i cilja, gdje se start i cilj nalaze u različitim potkartnim čvorovima. Ta je putanja jednaka putanji sastavljenoj od parcijalnih putanja određenih algoritmom D* između mosnih čvorova, jer su mosni čvorovi na mjestima kroz koja mora prolaziti optimalna putanja (u središtima uskih prolaza). Slika 3.5 prikazuje primjer loše pozicije mosnih čvorova. Parcijalne putanje naznačene su crtkanom linijom i vidljivo je odstupanje optimalne putnje od unije parcijalnih putanja.

3.3.2 Organizacija hijerarhije

U [19] je mosni čvor povezan sa čvorom raskrižja iz više razine hijerarhijske apstrakcije. Čvorovi raskrižja imaju ulogu skretnica na putanjama. Drugim riječima postoje definirana mjesta u kojima robot mora promijeniti smjer gibanja. Optimalnost putanje ovisi naravno o samom H-grafu. Ako čvor raskrižja predstavlja jedinu opciju prelaska iz jednog potkartnog čvora u drugi, tada je ta putanja optimalna. Međutim, ako je najdetaljnija razina složena tako da ona vjerno opisuje stvarni prostor gustom raspodjelom čvorova, odnosno njihovih koordinata, onda može postojati više mogućnosti. Prema tome, ovako određena putanja nije nužno optimalna budući da može postojati kraća putanja, koja prolazi nekim drugim čvorom. U ovome se radu razmatra graf na najdetaljnijoj razini sastavljen iz mrežaste karte zauzeća sa sigurnosnom maskom cijena.

U prethodnom je odlomku opisano optimalno postavljanje mosnih čvorova na granici između dvaju potkartnih čvorova tako da ih dijele oba potkartna čvora. Pokazano je da optimalne putanje kroz dva potkartna čvora moraju prolaziti mosnim čvorovima. Stoga se dalje ne koriste čvorovi raskrižja jer se putanje križaju u samim mosnim čvorovima. U tako definiranoj hijerarhiji smanjen je broj potrebnih čvorova na višim razinama, što dodatno ubrzava izvođenje algoritma FHD*. Međutim, mosni čvorovi povezuju potkartni čvor u kojem se nalaze s potkartnim čvorom više razine na način da se vežu sa samim sobom u razini iznad (višoj razini). Drugim riječima, mosni se čvorovi dupliciraju iz razine u kojoj su definirani, u višu razinu.

U nastavku se ponovo navode neke definicije H-grafa, koje su izmijenjene te neke nove definicije. Nasuprot definiciji potkartnih čvorova u odlomku 3.1.2, ovdje potkartni čvorovi nisu disjunktni podskupovi čvorova iz niže razine. Razmotrimo najnižu razinu apstrakcije. Mosni su čvorovi postavljeni na središta vrata između dviju soba, odnosno, postavljeni su na granicu između dva potkartna čvora tako da ih dijele oba potkartna čvora. Drugim riječima mosni čvor $m \in \mathcal{N}_D$ koji je na granici dvaju potkartnih čvorova $k, h \in \mathcal{N}_{D-1}$, nalazi se i u jednom i u drugom potkartnom čvoru, $m \in k$ i $m \in h$. Prema tome, mosni čvor ima dva potkartna čvora roditelja. Funkcija *roditelj* vratit će dva potkartna čvora, ako joj je argument mosni čvor. Međutim, u algoritmu hijerarhijskog planiranja putanje neće se koristiti funkcija *roditelj* nad mosnim čvorovima. Stoga mijenjamo samo domenu funkcije: $roditelj : \mathcal{N} \setminus \mathcal{M} \to \mathcal{K}$.

Duplicirani mosni čvor $m^i \in \mathcal{N}_i$, $(0 < i < D)$, jest ponovno stvoreni čvor u razini R_i takav da su koordinate čvora m^i jednake koordinatama *rubnog mosnog čvora* $m^{i+1} \in \mathcal{N}_{i+1}$ i postoji luk $a(m^i, m^{i+1}, w) \in \mathcal{A}$ gdje je $w = 0$. Duplicirani mosni čvor $m^i \in \mathcal{N}_i$

vezan je za one potkartne čvorove unutar istog potkartnog čvora roditelja $l \in \mathcal{N}_{i-1}$ u kojima se nalazi njegov izvorni mosni čvor m^{i+1}, odnosno vrijedi:

$$\begin{gathered}\text{ako } \exists k \in \mathcal{N}_i \text{ i } l \in \mathcal{N}_{i-1} \mid k, m^i \in l \text{ i } m^{i+1} \in k \text{ i } \exists a(m^i, m^{i+1}, w) \in \mathcal{A} \\ \text{tada } \exists a(m^i, k, w_x) \in \mathcal{A}.\end{gathered} \tag{3.11}$$

Duplicirani mosni čvor $m^i \in \mathcal{N}_i$, $(1 < i < D)$, unutar potkartnog čvora $l \in \mathcal{N}_{i-1}$ postaje *rubni mosni čvor* ako isti potkartni čvor l ne obuhvaća duboko u svojoj unutrašnjosti (gledajući po sve nižim razinama hijerarhijske apstrakcije do najniže) oba roditelja od izvornog mosnog čvora iz razine R_D. Rubni mosni čvor vezan je za samo jedan potkartni čvor unutar svog roditelja. Početni uvjet jest da su svi mosni čvorovi u razini R_D rubni mosni čvorovi. Rubni mosni čvor m može se duplicirati kroz više razina $R_D, R_{D-1}, \ldots, R_i$, označen odgovarajućim indeksima $m, m^{D-1}, \ldots, m^i$, gdje razina R_i, $(0 < i < D-1)$, sadrži potkartne čvorove koje spaja mosni čvor m^i. Tada su čvorovi $m^{D-1}, \ldots, m^{i+1}$ rubni mosni čvorovi, a čvor m^i duplicirani mosni čvor. Skup svih rubnih mosnih čvorova označavamo sa $\mathcal{M}_r$.

Sljedeći primjeri pokazuju da se prema navedenoj definiciji mosnog čvora vertikalni mosni čvorovi "ponašaju" isto kao i horizontalni.

Razmotrimo vertikalne mosne čvorove. Oni su postavljeni na mjesta vrata dizala, na granici između unutrašnjosti dizala i prostorije u kojoj se nalaze vrata dizala, a pojavljuju se kao rubni mosni čvorovi u razinama R_D, $R_{D-1}, \ldots, R_{i+1}$, u oblicima svojih "kopija", gdje razina R_i, $(0 < i < D-1)$, sadrži potkartne čvorove koji predstavljaju katove koji su povezani dizalom. Na ovoj se razini može nalaziti rubni mosni čvor (horizontalni) koji povezuje potkartne čvorove – zgrade. Primjerice, mosni čvor koji predstavlja vrata u prizemlju, povezuje prizemlje druge zgrade. Ovaj se mosni čvor nalazi (odnosno njegove "kopije") kao rubni mosni čvor u razinama R_D, $R_{D-1}, \ldots, R_i$, dakle u jednoj razini više nego vertikalni mosni čvorovi koji povezuju katove.

Navedimo dodatni primjer za horizontalne mosne čvorove. Mosni čvor m u razini R_D je rubni mosni čvor. Njegov duplicirani čvor m^{D-1} spaja dva potkartna čvora k i h na razini R_{D-1}. Međutim na razini R_{D-2} nalazi se potkartni čvor l koji sadrži čvor k, ali ne i h. Stoga je i čvor m^{D-1} rubni mosni čvor koji se duplicira u čvor m^{D-2} na razini R_{D-2}. Dupliciranje se nastavlja u višim razinama sve dok se ne pojavi potkartni čvor koji obuhvaća duboko u svojoj unutrašnjosti (rekurzivno izdvajanje čvora "dijete" od čvora "djeteta") oba potkartna čvora roditelja k i h od čvora m.

Rubni mosni čvorovi su "pravi" mosni čvorovi po izvornoj definiciji H-grafa. Ovdje navedena definicija mosnog čvora ne radi distinkciju između horizontalnih i vertikalnih mosnih čvorova. Prema tome vertikalno planiranje putanje bit će identično horizontalnom planiranju putanje.

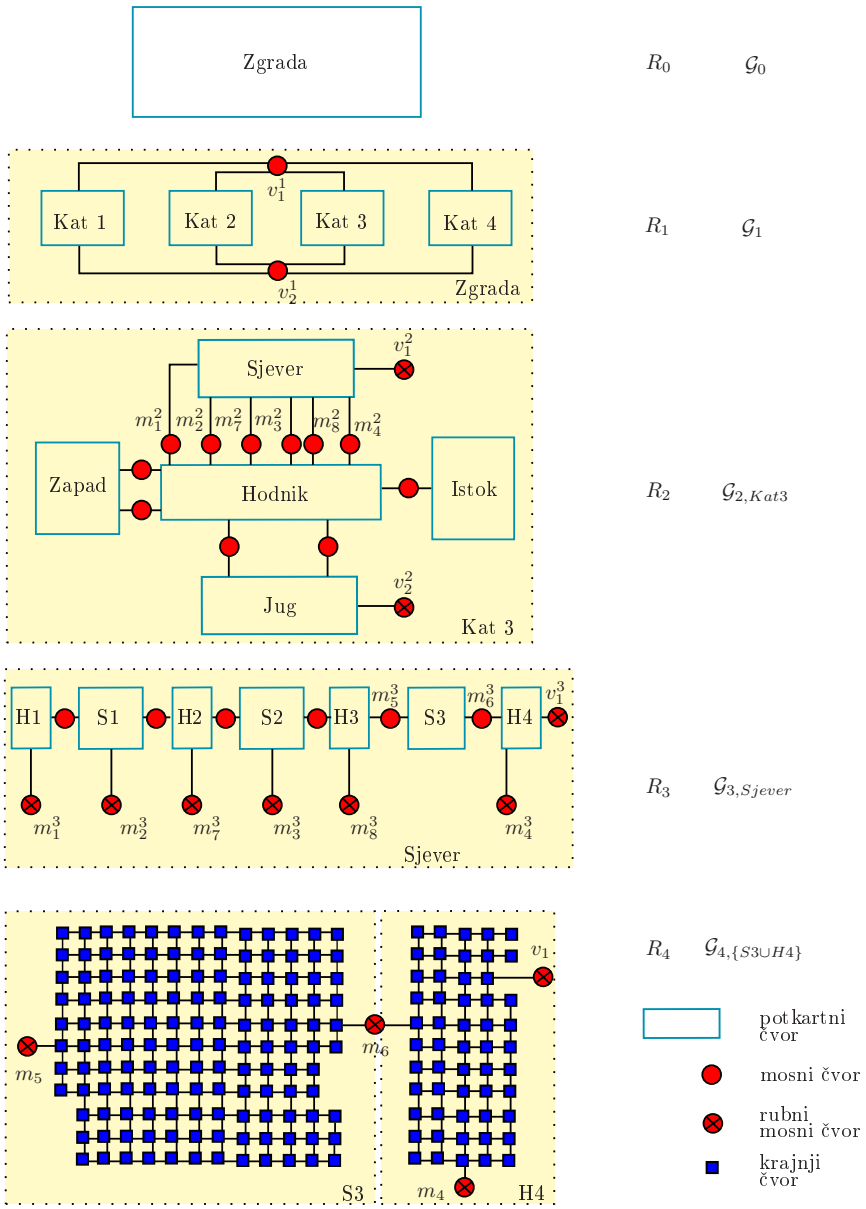
Na najnižoj razini nalaze se samo krajnji i mosni čvorovi, a na svim ostalim razinama samo potkartni i mosni čvorovi. Mosni čvorovi mogu biti na svim razinama osim na korijenskoj razini (R_0). Mosni čvorovi koji se nalaze u razini R_1 nastali su $(D-1)$-strukim dupliciranjem mosnih čvorova iz razine R_D. Startni i ciljni čvorovi mogu se

odabrati samo na najnižoj razini.

Lukovi na najnižoj razini ($\mathcal{A}_D$) definirani su prema definiciji bridova u poglavlju 2.2.2. Ako postoji brid $\{m,n\} \in \mathcal{E}$ s pripadajućom težinom $w_{m,n}$, tada je $a := a(m,n,w_{m,n})$.

Za razliku od prijašnje tri kategorije parcijalnih putanja, računamo samo tip 1 – putanja koja spaja dva mosna čvora unutar potkartnog čvora k. FHD* koristi mnogo manje materijaliziranih cijena u grafu od algoritma HD* pa je memorijski manje zahtjevan. Parcijalne se putanje određuju za svaki potkartni čvor između svih mogućih parova rubnih mosnih čvorova unutar jednog potkartnog čvora, počevši od najdetaljnije razine. Nakon što se odrede parcijalne putanje u jednoj razini one su osnova za računanje novih parcijalnih putanja u višoj razini. U višoj su razini dupliciranimosni čvorovi vezani za potkartne čvorove. Između svaka dva mosna čvora m_1^i, m_2^i između kojih postoji potkartni čvor $k \in \mathcal{N}_i$ na koji su vezani, postoji parcijalna putanja određena u nižoj razini unutar potkartnog čvora k, $\mathcal{T}(m_1^{i+1}, m_2^{i+1}, k)$. Ta putanja čini vezu čvorova m_1^i i m_2^i preko čvora k i određuje njihovu cijenu prijelaza kroz čvor k, $c(\mathcal{T})$. Pomoću tih (materijaliziranih) cijena, određuju se parcijalne putanje između svih mogućih parova rubnih mosnih čvorova koji se nalaze u istom potkartnom čvoru roditelju. Nove parcijalne putanje postaju osnova za višu razinu. Proces određivanja parcijalnih putanja završava kada se odrede putanje u zadnjoj razini s mosnim čvorovima - razina R_1. Zbog dupliciranja mosnih čvorova u višim razinama, parcijalna putanja koja spaja mosni čvor iz potkartnog čvora k s mosnim čvorom potkartnog čvora roditelja čvora k (tip 2) predstavlja istu parcijalnu putanju kao i putanja koja spaja dva mosna čvora iz "bratskih" potkartnih čvorova (tip 3). Računanje ovih kategorija putanja (tip 2 i tip 3) nije potrebno jer ih brzo određuje algoritam hijerarhijskog planiranja putanje – FHD* iz parcijalnih putanja (tip 1).

Razmotrimo ponovo primjer iz odlomka 3.1.4 prikazan na slici 3.2. Prema izmijenjenim definicijama H-grafa na slici 3.6 prikazan je modificirani primjer H-grafa. Pogledajmo primjer od najdetaljnije do najapstraktnije razine. Hodnici su zamijenjeni detaljnim prikazom svoje unutrašnjosti, kao i sobe, i nalaze se na najdetaljnijoj razini. Na razini R_4 nalaze se sve unutrašnjosti svih soba i hodnika (svih katova), a na slici je prikazana samo unutrašnjost sobe $S3$ i hodnika $H4$. Mosni čvor m_6 dijele i soba i hodnik. Svi mosni čvorovi na razini R_4 su rubni i dupliciraju se na razinu R_3. Tako se naznačeni mosni čvorovi m_4, m_5, m_6 i v_1 dupliciraju u čvorove m_4^3, m_5^3, m_6^3 i v_1^3. Na razini R_3 prisutni su i drugi mosni čvorovi koji su dupliciranii iz razine R_4, a sadržani su u sobama $S1$, $S2$ i hodnicima $H1$, $H2$, $H3$. Mosni čvorovi m_5^3 i m_6^3 nisu rubni čvorovi jer postoji čvor $Sjever$, koji obuhvaća njihove potkartne čvorove roditelje $H3$, $S3$ i $H4$. Oni povezuju svoje potkartne čvorove na razini R_3. Rubni mosni čvorovi iz razine R_3 dupliciraju se u razinu R_2 i tamo svi osim čvorova v_1^2 i v_2^2 povezuju svoj potkartni čvor s drugim potkartnim čvorovima na razini R_3. Rubni mosni čvorovi v_1^2 i v_2^2 dupliciraju se u čvorove v_1^1 i v_2^1 na razini R_1 i tamo povezuju svoje potkartne čvorove. Iz samih naziva potkartnih čvorova, jasno je da su v_1 i v_2 i svi njihovi dupliciranii čvorovi vertikalni mosni



Slika 3.6. *Modificirani primjer H-grafa zgrade s četiri kata.*

čvorovi. Vertikalni mosni čvorovi prisutni su kao i horizontalni u najdetaljnijoj razini. U primjeru se jedan vertikalni mosni čvor nalazi u hodniku $H4$, koji je unutar potkartnog čvora $Sjever$, a drugi vertikalni mosni čvor nalazi se u hodniku unutar potkartnog čvora Jug. Međutim, zanemarimo značenja naziva: na razini R_2 mogu svi potkartni čvorovi predstavljati katove, svi m čvorovi predstavljati dizala, a v čvorovi predstavljati izlaze iz cijelog kompleksa. Primjerice, čvor $Hodnik$ je garaža pod zemljom, koja se sa šest vertikalnih mosnih čvorova veže na suterensku parkirnu razinu $Sjever$ koja ima izlaz van cijelog kompleksa u v_1^2. Također se sa dva vertikalna mosna čvora veže na prizemnu

parkirnu razinu Jug, koja ima drugi izlaz van cijelog kompleksa u v_2^2. Dodatno, dva vertikalna mosna čvora vode u parkirnu razinu dublje pod zemljom $Zapad$, a jedan vertikalni mosni čvor odvojeno vodi u parkirnu razinu dublje pod zemljom $Istok$. Na razini R_1 izlazi v_1^1 i v_2^1 povezuju sve komplekse: garažu, šoping centar, *wellness* centar i kina multipleks.

Valja napomenuti da se u slučaju više zgrada ne moraju sve zgrade modelirati istim brojem hijerarhijskih apstrakcija. Pogledajmo sljedeći primjer: jedna zgrada ima četiri kata, svaki kat ima velik broj prostorija i hodnika koji se moraju grupirati u potkartne čvorove i takva zgrada zahtijeva razine $R_0, \ldots, R_5$. Druga zgrada je prizemnica sa samo nekoliko soba i jednim hodnikom i ona zahtijeva razine R_0, R_1, R_2. Povezivanjem tih dviju zgrada u zajedničku hijerarhijsku strukturu dobivamo razine $R_0, \ldots, R_6$, od kojih razine R_4, R_5, R_6 za drugu zgradu ostaju neiskorištene. Prema tome, problem planiranja putanje između ove dvije zgrade imat će start na jednoj razini hijerarhijske apstrakcije, a cilj na drugoj. U slučaju modeliranja samo jedne zgrade, pretpostavlja se da su svi katovi jednake strukture i da svaki kat koristi isti broj razina hijerarhijske apstrakcije.

3.3.3 Usmjeravanje hijerarhijskog pretraživanja oko optimalne putanje

Velika prednost hijerarhije opisane u prethodnom odlomku jest u hijerarhijskom planiranju putanje. Pri pretraživanju čvorova na višim razinama hijerarhijske apstrakcije nije potrebno odmotavati potkartni čvor, kao što se radi u algoritmu HD*. U organizaciji hijerarhije kakvu koristi algoritmu HD* čvor križanja povezan je s potkartnim čvorom i potrebno je razmotati potkartni čvor, odnosno prebaciti se u nižu razinu da se pronađu mosni čvorovi od tog potkartnog čvora i razmotre njihove parcijalne putanje. U predloženoj organizaciji hijerarhije mosni su čvorovi ponovljeni u višoj razini, gdje se nalazi njihov potkartni čvor i postavljene su veze između čvorova tako da se svo nužno pretraživanje odvija u jednoj razini, a dalje se nastavlja u višim razinama bez vraćanja na nižu ranizu.

U [18] je korištena heuristika prema cilju, a čvorovi se pretražuju redom prema vrijednosti funkcije $f = c + g$, gdje je c cijena putanje koja se razmatra, a g procjena cijene ostatka putanje do cilja. Cijena koja se procjenjuje heurističnom funkcijom g može se zamijeniti egzaktnom cijenom putanje g^* do cilja. Mobilni robot na putanji do cilja mijenja svoju trenutnu (startnu) poziciju, odnosno startni čvor se za algoritam planiranja putanje mijenja. U dinamičkim prostorima korisnije je koristiti heuristiku prema startu, budući da se dio prostora ispred mobilnog robota mijenja, a onaj oko zadanog cilja ostaje nepromijenjen. Dinamičko planiranje algoritmom A* sporije je nego algoritmom FD*. U izvedbi algoritma A* ne postoji izračunata putanja do cilja iz svih čvorova u pretraženom području, kao što postoje u izvedbi algoritma FD*. Algoritam FD* koristi sve izračunate putanje od zadnjeg izvođenja algoritma u novim izvođenjima i zato je vremenski učinkovit, dok A* može koristiti samo jednu putanju – inicijalna putanja. Pogledajmo svojstvo funkcije f kod algoritama A* i FD*. Budući da heuristika

mora biti dopustiva i konzistentna (odlomak 2.3.1), prema lemi 2.3 vrijedi:

$$f(S) \leq f(n) \leq f(G), \tag{3.12}$$

za svaki proširen čvor n algoritmom A*, odnosno za svaki proširen čvor n algoritmom FD*

$$f(G) \leq f(n) \leq f(S). \tag{3.13}$$

Budući da su redom pretraživani čvorovi koji imaju najmanju vrijednost funkcije f, u procesu dinamičkog planiranja algoritam FD* odabire čvorove bliže cilju, a algoritam A* odabire čvorove bliže startu. Prema tome, dinamičko planiranje algoritmom A* počinje startnim čvorom, a dinamičko planiranje algoritmom FD* počinje čvorom koji je postao zauzetim. Lokalno preusmjeravanje pokazivača oko čvora s preprekom mnogo je brže od pretraživanja čvorova od startnog čvora.

3.3.4 Inicijalno planiranje putanje

Pretpostavka je da se start (S) i cilj (G) ne nalaze u istom potkartnom čvoru, inače se radi o običnom algoritmu FD*. Neka je R_i $(0 \leq i < D-1)$ razina u kojoj je definiran potkartni čvor koji obuhvaća i start i cilj u svojim potkartnim čvorovima u najvećoj dubini hijerarhije. Nazovimo taj čvor zajedničkim potkartnim čvorom i označimo ga s k^i_{SG} (algoritam 3.6).

Algoritam 3.6: zajednički-potkartni-čvor(S, G))

```
1:  k_S ← roditelj(S)
2:  k_G ← roditelj(G)
3:  dok k_S ≠ k_G
4:      k_S ← roditelj(k_S)
5:      k_G ← roditelj(k_G)
6:  kraj
7:  vrati k_S
```

Pretraživanje počinje od najniže razine R_D od starta i od cilja paralelno (algoritmi traži-od-starta i **traži-od-cilja**, 3.7). Koristimo skup $\mathcal{O}$ u koji se dodaju čvorovi koji se trenutno pretražuju, a iz kojeg se oduzimaju pretraženi čvorovi. Od starta se pretražuju čvorovi redom prema rastućoj vrijednosti funkcije h, gdje je h akumulirana cijena putanje od starta. Analogno, od cilja se pretražuju čvorovi redom prema rastućoj vrijednosti funkcije g, gdje je g akumulirana cijena putanje od cilja. Pretražuju se samo čvorovi unutar istog potkartnog čvora roditelja od starta ($k_S^{D-1} = roditelj(S)$), odnosno od cilja ($k_G^{D-1} = roditelj(G)$, linija 7 u algoritmu 3.7). Pretraženim čvorovima pamte se akumulirane cijene g i h i pamte se pokazivači $b_G(\cdot)$ i $b_S(\cdot)$ pomoću kojih se može rekonstruirati optimalnu putanju iz bilo kojeg pretraženog čvora do cilja (starta) (za najdetaljniju razinu to su linije 9-13 u algoritmu 3.7). Pretraživanje u razini R_D traje sve dok se ne dohvate svi rubni mosni čvorovi u startnom, odnosno ciljnom potkartnom

Algoritam 3.7: traži-od-cilja(G)

```
1:  O ← G, g(G) ← 0, k ← roditelj(G)   //Inicijalizacija
2:  dok O ≠ ∅
3:      g_min ← min{g(n) | n ∈ O \ M_r}
4:      ako n* ≠ ∅   //za n* vrijedi g(n*) = g_min
5:          o ← n*
6:          O ← O \ o
7:          za ∀n ∈ k tako da ∃a(o, n, w_on) ∈ A
8:              ako n ∉ K
9:                  ako n ∉ O ili g(n) > g(o) + w_on
10:                     O ← O ∪ n
11:                     g(n) ← g(o) + w_on
12:                     b_G(n) ← o
13:                 kraj
14:             inače
15:                 r ← r ∈ n | ∃a(r, o, w) ∈ A
16:                 za ∀p ∈ k tako da ∃a(n, p, w_x) ∈ A
17:                     m ← m ∈ n | ∃a(m, p, w) ∈ A
18:                     T ← T(r, m, n)
19:                     ako p ∉ O ili g(p) > g(o) + c(T)
20:                         O ← O ∪ p
21:                         g(p) ← g(o) + c(T)
22:                         b_G(p) ← o
23:                         b_G(n) ← o  //radi rekonstrukcije parcijalnih putanja
24:                     kraj
25:                 kraj
26:             kraj
27:         kraj
28:     inače  //ostali su samo rubni mosni čvorovi
29:         za ∀m ∈ O
30:             O ← O \ m
31:             k ← roditelj(k)
32:             ako ∃n ∈ k tako da ∃a(n, m, w) ∈ A
33:                 O ← O ∪ n
34:                 g(n) ← g(m) + w   //w = 0
35:                 b_G(n) ← m
36:             kraj
37:         kraj
38:     kraj
39: kraj
```

čvoru roditelju, i samo oni ostanu u skupu $\mathcal{O}$ (linije 3 i 28). Tada postoji optimalna putanja iz svakog rubnog mosnog čvora iz potkartnog čvora k_S^{D-1} do starta i analogno, iz svakog rubnog mosnog čvora iz potkartnog čvora k_G^{D-1} do cilja. Rubni mosni čvorovi oduzimaju se zadnji iz skupa $\mathcal{O}$.

Pretraživanje se prebacuje u višu razinu (R_{D-1}) tako što se rubni mosni čvorovi zamijene njihovim dupliciranim mosnim čvorovima iz više razine i odredi se novi pot-

kartni čvor roditelj (linije 29-37). Označimo te roditelje sa $k_S^{D-2} = roditelj(k_S^{D-1})$ i $k_G^{D-2} = roditelj(k_G^{D-1})$. Valja napomenuti da je akumulirana cijena dupliciranog mosnog čvora m^i jednaka akumuliranoj cijeni mosnog čvora m^{i+1}, budući da po definiciji dupliciranog mosnog čvora postoji luk $a(m^i, m^{i+1}, w)$ i ima težinu $w = 0$. Dalje se pretražuju svi čvorovi na isti način kao i na najdetaljnijoj razini, osim što ovdje postoje i potkartni čvorovi (linije 15-25). Oni se pretražuju tako što se koriste cijene parcijalnih putanja iz niže razine. Između svaka dva mosna čvora m_1^i, m_2^i između kojih postoji potkartni čvor $n \in \mathcal{N}_i$ s kojim su vezani, postoji parcijalna putanja određena u nižoj razini unutar potkartnog čvora n, $\mathcal{T}(m_1^{i+1}, m_2^{i+1}, n)$. Ta putanja čini vezu čvorova m_1^i i m_2^i preko čvora n i određuje njihovu cijenu prijelaza kroz čvor m, $c(\mathcal{T})$. U skup $\mathcal{O}$ ne ubacuju se potkartni već samo mosni čvorovi. Pokazivači su postavljeni tako da se kasnije može rekonstruirati parcijalna putanja između dvaju mosnih čvorova (algoritam 3.8). Rubni mosni čvorovi vezani su samo s jednim potkartnim čvorom tako da će oni biti završni čvorovi u pretraživanju na svakoj razini (linije 29-37). Postupak se dalje ponavlja kroz više razine hijerarhijske apstrakcije sve dok se ne dosegne razina R_0. Pretraživanje se mora nastaviti i iznad razine sa zajedničkim potkartnim čvorom k_{SG}^i jer se može dogoditi da postoje putanje kroz "nebratske" potkartne čvorove (koji nisu sadržani u k_{SG}^i) zbog strukture hijerarhije.

Algoritam 3.8: ubaci-parcijalne-putanje($\mathcal{P}$)

```
1:  za ∀m ∈ P ∩ M
2:      i ← i | m = P[i]
3:      P1 ← P[1, ..., i]
4:      P2 ← P[i + 1, ..., |P|]
5:      p ← P1[|P1|]
6:      o ← P2[1]
7:      ako je p, o ∈ M i razina(p) = razina(o) i ∃n, a(n, o, wx) ∈ A
        tako da b(n) = o
8:          r ← r ∈ n | ∃a(r, o, w) ∈ A
9:          m ← m ∈ n | ∃a(m, p, w) ∈ A
10:         T ← T(r, m, n)
11:         P ← P1 ∪ T ∪ P2
12:     kraj
13: kraj
14: vrati P
```

Pseudokod inicijalnog planiranja algoritma FHD* dan je algoritmom 3.9. Po završetku pretraživanja i od starta i od cilja, traži se mosni čvor $m \in k_{SG}^i$, koji ima najmanju cijenu $f(m) = g(m) + h(m)$ (algoritam 3.9, linije 4-5). Čvor n^* koji ima najmanju cijenu f nalazi se na optimalnoj putanji. Od n^* do cilja određena je putanja $\mathcal{P}_G$, i analogno je od istog čvora do starta određena putanja $\mathcal{P}_S$ (linije 7-12 i 14-16). Putanja se djelomično sastoji od niza mosnih čvorova između kojih se ubacuju parcijalne putanje (algoritam 3.8). Ukupna putanja $\mathcal{P}(S, G)$ odredi se unijom putanja $\mathcal{P}_S$ i $\mathcal{P}_G$, ali tako da se obrne redoslijed čvorova u putanji $\mathcal{P}_S$.

Algoritam 3.9: inicijalno-planiranje-FHD*(S, G)

```
1:  za ∀n ∈ 𝒩, g(n) ← ∞  //Inicijalizacija
2:  traži-od-cilja(G)  //odredi svakom čvoru g, b_G
3:  traži-od-starta(S)  //odredi svakom čvoru h, b_S
4:  k_SG ←zajednički-potkartni-čvor(S, G)
5:  f_min ← min{g(n) + h(n) | n ∈ ℳ ∩ k_SG}
6:  ako f_min < ∞
7:      𝒫_S = 𝒫(n*, S)   //za n* vrijedi g(n*) + h(n*) = f_min
8:          𝒫_S[1] ← n*
9:          za i = 2, ..., |𝒫_S|
10:             𝒫_S[i] ← b_S(𝒫_S[i − 1])
11:             g(𝒫_S[i]) ← g(𝒫_S[i − 1]) + h(𝒫_S[i − 1]) − h(𝒫_S[i])
12:         kraj
13:     𝒫_S ←ubaci-parcijalne-putanje(𝒫_S)
14:     𝒫_G = 𝒫(n*, G):
15:         𝒫_G[1] ← n*
16:         𝒫_G[i] ← b_G(𝒫_G[i − 1]), i = 2, ..., |𝒫_G|
17:     𝒫_G ←ubaci-parcijalne-putanje(𝒫_G)
18:     𝒫 = 𝒫(S, G):
19:         𝒫[1] ← 𝒫_S[|𝒫_S|]
20:         𝒫[i] ← 𝒫_S[|𝒫_S| − i + 1], i = 2, ..., |𝒫_S|
21:         𝒫 ← 𝒫 ∪ 𝒫_G
22: inače
23:     𝒫 ← ∅
24: kraj
25: vrati 𝒫
```

Za daljnje izračune algoritma potrebne su cijene g i funkcija k koja pamti staru vrijednost cijene g. Čvorovi koji se nalaze na putanji $\mathcal{P}_S$ nemaju određene vrijednosti g, ali se one jednostavno određuju iz vrijednosti h i poznate vrijednosti g na početnom čvoru (linija 11 algoritma 3.9).

3.3.5 Dinamičko planiranje putanje

Proces dinamičkog planiranja sličan je dinamičkom planiranju algoritmom FD* u jednorazinskom grafu. Pseudokod dinamičkog planiranja putanje algoritma FHD* dan je algoritmom 3.10. Počinje od najniže razine. Čvor Z koji predstavlja polje koje je postalo zauzeto ažurira se prvi i propagira porast cijene prema trenutnoj poziciji robota - startu (algoritam 3.15). Čvorovi se ažuriraju tako što se dodaju u skup $\mathcal{O}$ (algoritam 3.11). Tada im se određuju vrijednosti funkcija k, g, f. Heuristika se računa prema trenutnoj poziciji robota – S.

Pretraživanje se odvija u potkartnom čvoru roditelju od čvora Z, nazovimo ga k_Z^{D-1}. Ako je čvor Z mosni čvor, tada on ima dva potkartna čvora roditelja i pretraživanje se odvija u čvoru koji je na putanji bliži startnom čvoru. Iz skupa $\mathcal{O}$ oduzima se čvor o koji ima najmanju cijenu funkcije $f = k + g$, a razmatraju se svi čvorovi koji su lukom

Algoritam 3.10: dinamičko-planiranje-FHD*(S, G, Z)

```
1:  traži-od-prepreke(Z, S)  //odredi čvorovima g, b_G, f, k
2:  spoji-mosne-čvorove(S)  //odredi čvorovima h, b_S
3:  k_S ← roditelj(S)
4:  f_min ← min{g(n) + h(n) | n ∈ M ∩ k_S}
5:  ako f_min < ∞
6:      P_S = P(n*, S)   //za n* ∈ M ∩ k_S vrijedi g(n*) + h(n*) = f_min
7:          P_S[1] ← n*
8:          za i = 2, ..., |P_S|
9:              P_S[i] ← b_S(P_S[i − 1])
10:             g(P_S[i]) ← g(P_S[i − 1]) + h(P_S[i − 1]) − h(P_S[i])
11:         kraj
12:     P_S ←ubaci-parcijalne-putanje(P_S)
13:     P_G = P(n*, G):
14:         P_G[1] ← n*
15:         P_G[i] ← b_G(P_G[i − 1]), i = 2, ..., |P_G|
16:     P_G ←ubaci-parcijalne-putanje(P_G)
17:     P = P(S, G):
18:         P[1] ← P_S[|P_S|]
19:         P[i] ← P_S[|P_S| − i + 1], i = 2, ..., |P_S|
20:         P ← P ∪ P_G
21: inače
22:     P ← ∅
23: kraj
24: vrati P
```

Algoritam 3.11: dodaj-FHD*(n, g_{min})

```
1:  ako n ∈ O
2:      k(n) ← min{g(n), g_min}
3:  inače
4:      k(n) ← min{k(n), g_min}
5:  kraj
6:  g(n) ← g_min
7:  h(n) ← heuristika(n, S)
8:  f(n) ← k(n) + h(n)
9:  O ← O ∪ n
```

Algoritam 3.12: smanji-cijenu(o, n, w)

```
1:  ako k(o) < g(o) i [(h(n) + g(n)), g(n)] ≤ [f(o), k(o)] i g(o) > g(n) + w
2:      g(o) ← g(n) + w
3:      b_G(o) ← n
4:  kraj
```

Algoritam 3.13: širi-smanjenje(o, n, w)

```
1:  ako k(o) = g(o) i (b_G(n) = o i g(n) ≠ g(o) + w) ili (g(n) > g(o) + w)
2:      dodaj-FHD*(n, (g(o) + w))
3:      b_G(n) ← o
4:  kraj
```

Algoritam 3.14: širi-porast(o, n, w)

```
1:  ako k(o) < g(o)
2:      ako (b_G(n) = o i g(n) ≠ g(o) + w)
3:          dodaj-FHD*(n, (g(o) + w))
4:      inače ako (b_G(n) ≠ o i g(n) > g(o) + w) i o ∉ O
5:          dodaj-FHD*(o, g(o))
6:      inače ako (b_G(n) ≠ o i g(o) > g(n) + w) i n ∉ O
        i [f(o), k(o)] < [(h(n) + g(n)), g(n)]
7:          dodaj-FHD*(n, g(n))
8:      kraj
9:  kraj
```

vezani za čvor o. S obzirom na odnos cijena $k(o)$ i $g(o)$ tom se čvoru može smanjiti cijena (algoritam 3.12), on može širiti smanjenje cijene drugim čvorovima (algoritam 3.13) ili može širiti porast cijene drugim čvorovima (algoritam 3.14). Te tri funkcije osnovne su funkcije algoritma FD*.

Rubni mosni čvorovi obrađuju se zadnji, jer su oni povezani s višom razinom. Njihovim dupliciranim mosnim čvorovima u višoj razini samo se prepišu vrijednosti cijena izvornih mosnih čvorova (algoritam 3.15, linije 25-33). Ako su u skupu $\mathcal{O}$ ostali samo rubni mosni čvorovi, u potkartnom čvoru ne postoji putanja i pretraživanje se nastavlja u višoj razini unutar potkartnog čvora roditelja k_Z^{D-2}. Proces dinamičkog planiranja na višoj razini identičan je dinamičkom planiranju u najnižoj razini, osim što se računa s materijaliziranim cijenama između dvaju mosnih čvorova, kao i u inicijalnom izračunu (algoritam 3.15, linije 14-21). U višim se razinama također rubni mosni čvorovi oduzimaju zadnji iz skupa $\mathcal{O}$, jer su oni povezani s višom razinom.

Pretraživanje traje dokle god ima čvorova u skupu $\mathcal{O}$. To čini razliku u odnosu na algoritam FD*, koji ima uvjet zaustavljanja ako cijena startnog čvora postane manja od najmanje cijene čvorova iz skupa $\mathcal{O}$. Ovdje se ne može koristiti taj uvjet jer se ne vraćamo na najnižu razinu u kojoj se nalazi start.

Zadnji je korak dinamičkog planiranja algoritmom FHD* vraćanje u najnižu razinu radi povezivanja starta s najboljim mosnim čvorom iz njegovog potkartnog čvora. Mosni čvorovi iz potkartnog čvora koji sadrži start nemaju putanju do starta (start je trenutna pozicija robota koja nije jednaka prvotnom startu). Međutim oni se moraju odrediti da bi se osigurala optimalnost ukupne putanje. Najučinkovitije je odrediti sve putanje u jednom prolazu pretraživanja. Pretraži se cijeli potkartni čvor koji sadrži start D* algoritmom počevši od starta. Po završetku pretraživanja, iz svakog čvora (unutar startovog

Algoritam 3.15: traži-od-prepreke(Z, S)

```
1:  dodaj-FHD*(Z, g(Z))
2:  k ← roditelj(Z)
3:  dok O ≠ ∅
4:      f_min ← min{f(n) | n ∈ O \ M_r}
5:      ako n* ≠ ∅  //za n* vrijedi f(n*) = f_min
6:          o ← n*
7:          O ← O \ o
8:          za ∀n ∈ k tako da ∃a(o, n, w_on) ∈ A
9:              ako n ∉ K
10:                 smanji-cijenu(o, n, w_on)
11:                 širi-smanjenje(o, n, w_on)
12:                 širi-porast(o, n, w_on)
13:             inače
14:                 r ← r ∈ n | ∃a(r, o, w) ∈ A
15:                 za ∀p ∈ k tako da ∃a(n, p, w_x) ∈ A
16:                     m ← m ∈ n | ∃a(m, p, w) ∈ A
17:                     T ← T(r, m, n)
18:                     smanji-cijenu(o, p, c(T))
19:                     širi-smanjenje(o, p, c(T))
20:                     širi-porast(o, p, c(T))
21:                 kraj
22:             kraj
23:         kraj
24:     inače //ostali su samo rubni mosni čvorovi
25:         za ∀m ∈ O
26:             O ← O \ m
27:             k ← roditelj(k)
28:             ako ∃n ∈ k tako da ∃a(n, m, w) ∈ A
29:                 O ← O ∪ {n}
30:                 k(n) ← k(m), g(n) ← g(m), f(n) ← f(m)  //Prepisivanje
31:                 b_G(n) ← m
32:             kraj
33:         kraj
34:     kraj
35: kraj
```

potkartnog čvora) postoji putanja i cijena putanje do starta određena pokazivačima b_S i cijenom h (algoritam 3.16).

Na kraju se traži najbolji mosni čvor iz startovog potkartnog čvora, koji ima najmanju cijenu $f = g + h$ (algoritam 3.10, linije 3-4). Čvor koji ima minimalnu cijenu f nalazi se na optimalnoj putanji. Putanja se odredi na sličan način kao i u inicijalnom pretraživanju (linije 6-20).

Algoritam 3.16: spoji-mosne-čvorove(S)

```
1:  k ← roditelj(S)
2:  ∀n ∈ k, h(n) ← ∞  //Inicijalizacija
3:  O ← S
4:  h(S) ← 0
5:  dok O \ M_r ≠ ∅:
6:      h_min ← min{h(n) | n ∈ O \ M_r}
7:      o ← n*   //za n* vrijedi h(n*) = h_min
8:      O ← O \ o
9:      za ∀n ∈ k tako da ∃a(o, n, w_on) ∈ A:
10:         ako n ∉ O ili h(n) > h(o) + w_on
11:                 O ← O ∪ n
12:                 h(n) ← h(o) + w_on
13:                 b_S(n) ← o
14:         kraj
15:     kraj
16: kraj
```

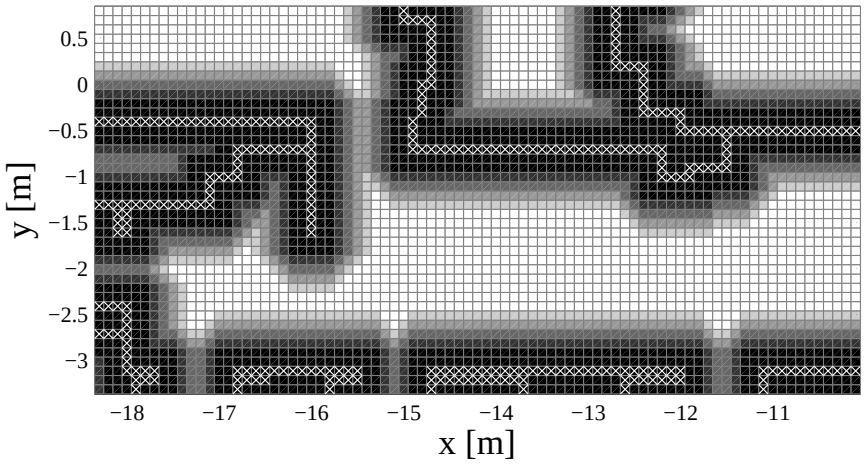
3.4 Automatska izgradnja hijerarhijske karte

Učinkovit pristup automatskoj izgradnji hijerarhijske apstrakcije prostora predložen je u [112]. Pristup je zasnovan na optimalno postavljenim mosnim čvorovima i izgrađuje kartu opisanu u odlomku 3.3.2. Konzistentnost hijerarhijske karte zajamčena je izgradnjom apstraktnih razina iz mrežaste karte zauzeća. Proces izgradnje hijerarhijske karte neovisan je o promjenama u prostoru i izvodi se samo jednom, inicijalno, prije početka planiranja putanje. Promjene u prostorima koje robot otkriva svojim senzorima pohranjuju se u mrežastoj karti zauzeća, dakle u najnižoj razini, a dalje se na učinkovit način korigiraju cijene čvorova na višim razinama algoritmom hijerarhijskog planiranja putanje (što je detaljno opisano u odlomku 3.3.3). Optimalnost izračunate putanje na takvom grafu osigurana je optimalnim pozicijama mosnih čvorova između kojih se računaju parcijalne putanje.

Predloženi algoritam određuje optimalne pozicije mosnih čvorova i izdvaja sobe iz najdetaljnije hijerarhijske apstrakcije unutarnjeg prostora – mrežaste karte zauzeća sa sigurnosnom maskom cijena (poglavlje 2.2.2). Sigurnosna maska cijena daje fini prijelaz cijena od prepreka prema slobodnom prostoru. Na takvoj karti algoritam radi sljedeće operacije u tri koraka: (i) izolira uske prolaze koji imaju važan zadatak u koordiniranju više robota, (ii) određuje pozicije mosnih čvorova na osnovi uskih prolaza i (iii) određuje grupacije slobodnih polja nazvanih *sobe* ograđenih preprekama i uskim prolazima od drugih soba. Sobe čine potkartne čvorove u višoj razini hijerarhije. Dalje se nastavlja jednostavan proces grupiranja potkartnih čvorova u nove potkartne čvorove idući prema apstraktnijim razinama hijerarhije.

3.4.1 Izdvajanje uskih prolaza

Prvi korak algoritma je pronalaženje uskih prolaza iz mrežaste karte zauzeća sa sigurnosnom maskom cijena. Slika 3.7 predstavlja detaljan opis dijela hodnika. Stvarne



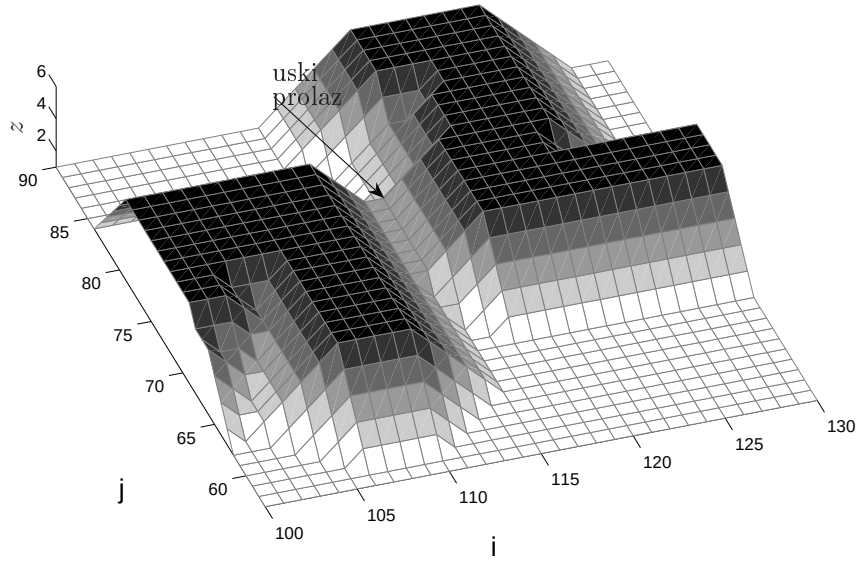
Slika 3.7. *Dio mrežaste karte zauzeća sa sigurnosnom maskom cijena na razini R_D.*

koordinate polja označene su na osima x i y. Radi jednostavnosti zapisa realna koordinata prostora $(x, y) \in \mathbb{R}^2$ pretvara se u cjelobrojne koordinate mrežaste karte zauzeća $(i, j) \in \mathbb{N}^2$:

$$(i, j) := (\lceil \frac{x - x_0}{e_{polje}} \rceil, \lceil \frac{y - y_0}{e_{polje}} \rceil), \tag{3.14}$$

gdje je e_{polje} duljina stranice polja, a x_0 i y_0 najmanje koordinate prostora (ishodište je lijevi donji ugao karte prostora). Prema tome, polje $n \in \mathcal{K}$ ima realne koordinate centra $c_n \equiv (x_n, y_n) \in \mathbb{R}^2$, a cjelobrojne $(i_n, j_n) \in \mathbb{N}^2$. Funkcija zauzetosti $z : \mathcal{K} \to \{1, 2, \ldots, z_{max}\}$ ovdje je predstavljena funkcijom dviju varijabli $z : \mathbb{N}^2 \mapsto \{1, 2, \ldots, z_{max}\}$, gdje je $z_{max} = M_c + 2$ maksimalna vrijednost sigurnosne cijene određena parametrom širine sigurnosne maske cijena M_c koji utječe na odmaknutost putanje od prepreka i može se odrediti kao što je opisano u poglavlju 5.3.3. Najmanja vrijednost funkcije z je u poljima izvan sigurnosne maske cijena, $z(\cdot, \cdot) = 1$, a najveća vrijednost u zauzetim poljima, $z(\cdot, \cdot) = z_{max}$. Trodimenzionalni prikaz funkcije z dan je na slici 3.8 s naznačenim uskim prolazom. Uski prolaz definiran je kao skup polja u kojima je prisutan lokalni minimum funkcije z, a vrijednost funkcije z je veća od 1. Najveća širina uskog prolaza jednaka je dvijema širinama sigurnosne maske cijena. Pri određivanju minimuma funkcije z dopuštamo da najviše dva susjedna polja poprimaju minimalnu vrijednost. Minimumi se određuju za $\forall (i, j) \in \mathbb{N}^2$ prema vrijednostima druge derivacije po varijabli i i j zasebno:

$$\begin{aligned} z_i''(i, j) &= z(i - 1, j) - 2z(i, j) + z(i + 1, j), \\ z_j''(i, j) &= z(i, j - 1) - 2z(i, j) + z(i, j + 1). \end{aligned} \tag{3.15}$$



Slika 3.8. *Dio prostora u 3D prikazu s vrijednostima funkcije zauzetosti z na z-osi.*

Neka je $\mathcal{U}$ skup čvorova čije se koordinate nalaze u uskom prolazu. Čvor $n \in \mathcal{N}$ koordinata (i_n, j_n) pridružuje se skupu $\mathcal{U}$ ako njegove koordinate ispunjuju sljedeće uvjete:

$$\begin{aligned} &\text{za } \forall n \in \mathcal{N} \text{ koordinata } (i_n, j_n) \\ &\text{ako je } z_i''(i_n, j_n) > 0 \text{ i } |z(i_n - 2, j_n) - z(i_n + 2, j_n)| \leq 1 \\ &\text{ili } z_j''(i_n, j_n) > 0 \text{ i } |z(i_n, j_n - 2) - z(i_n, j_n + 2)| \leq 1, \\ &\text{tada } \mathcal{U} \leftarrow \mathcal{U} \cup n. \end{aligned} \tag{3.16}$$

Dobiveni skup $\mathcal{U}$ razdjeljuje se na U disjunktnih podskupova. Svaki podskup $\mathcal{U}_k$, $k = 1, \ldots, U$, opisuje jedan uski prolaz u prostoru. Svaki element skupa $\mathcal{U}$ raspoređuje se u odgovarajući podskup $\mathcal{U}_k$. Početno je $\mathcal{U}_U \leftarrow \{n\} \in \mathcal{U}$, $\mathcal{U} \leftarrow \mathcal{U} \setminus \{n\}$ i $U = 1$, a dalje se rješava iterativno sljedeći izraz:

$$\begin{aligned} &\text{ako } \exists m \in \mathcal{U} \text{ i } n \in \mathcal{U}_U \text{ tako da } \{m, n\} \in \mathcal{E} \\ &\text{tada } \mathcal{U}_U \leftarrow \mathcal{U}_U \cup \{m\} \text{ i } \mathcal{U} \leftarrow \mathcal{U} \setminus \{m\} \\ &\text{inače } U \leftarrow U + 1, \quad \mathcal{U}_U \leftarrow \{n\} \in \mathcal{U}, \quad \mathcal{U} \leftarrow \mathcal{U} \setminus \{n\}. \end{aligned} \tag{3.17}$$

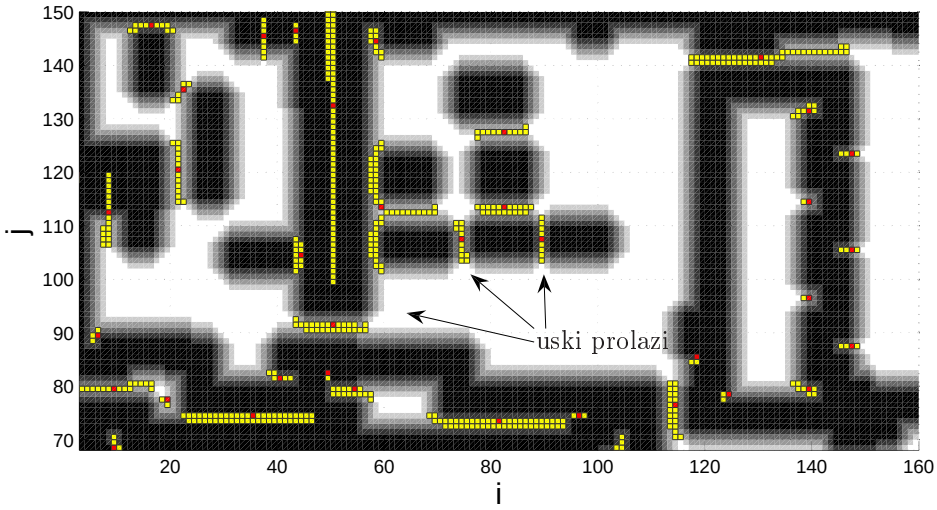
Postupak (3.17) ponavlja se sve dok nije $\mathcal{U} = \emptyset$. Svakom skupu $\mathcal{U}_k$, $k = 1, \ldots, U$, određuje se središnji čvor, označen $s_k \in \mathcal{U}_k$, čije su koordinate u središtu uskog prolaza. Središte prolaza određuje se kao bilo koja najbliža koordinata (može ih biti više) aritmetičkoj sredini $(x_s, y_s) \in \mathbb{R}^2$ svih koordinata čvorova iz $\mathcal{U}_k$. Središnji čvor s_k određuje

se kako slijedi:

$$(x_s, y_s) := \frac{1}{|\mathcal{U}_k|} \sum_{\substack{n \\ \text{u.u.}}} (i_n, j_n) \quad n \in \mathcal{U}_k \tag{3.18}$$

$$s_k \in \underset{\substack{n \\ \text{u.u.}}}{\arg\min} \|(x_s, y_s) - (i_n, j_n)\| \quad n \in \mathcal{U}_k \tag{3.19}$$

Moguće pozicije mosnih čvorova su u središnjim koordinatama, odnosno s_k može biti mosni čvor ako ispunjuje određene uvjete drugog koraka algoritma opisanog u odlomku 3.4.2. Slika 3.9 prikazuje uske prolaze u dijelu prostora predstavljenim mrežastom kartom zauzeća sa sigurnosnom maskom cijena.



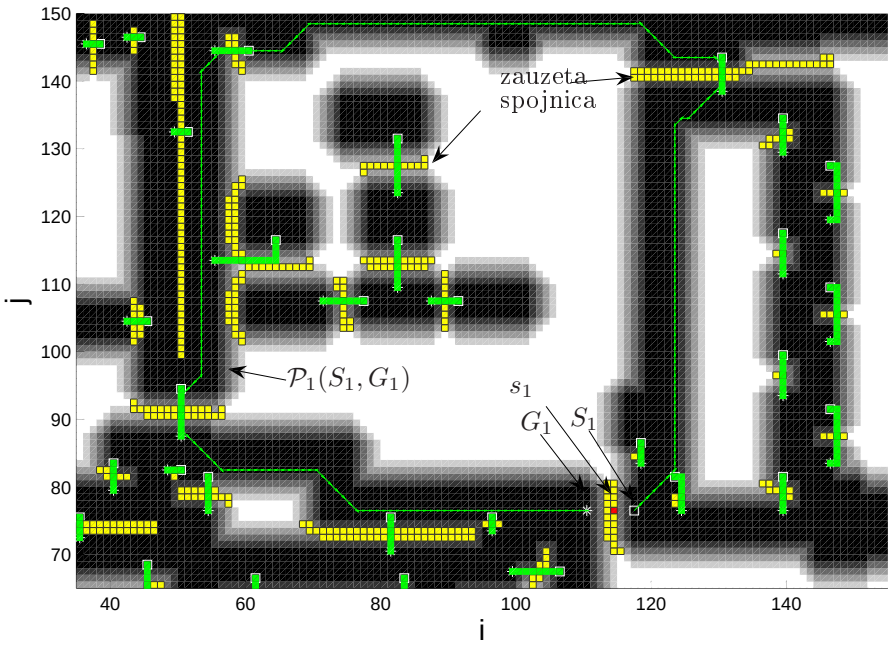
Slika 3.9. *Uski prolazi u dijelu prostora predstavljenim mrežastom kartom zauzeća sa sigurnosnom maskom cijena.*

3.4.2 Određivanje pozicije mosnih čvorova

Cilj je ovog koraka algoritma odrediti koje od središnjih koordinata uskih prolaza čine koordinate mosnih čvorova, odnosno odrediti skup mosnih čvorova $\mathcal{M}$. Mosni čvor mora biti na granici između dvaju potkartnih čvorova, odnosno, u našem slučaju, dvije sobe. Polja jedne sobe odvojena su od polja drugih soba preprekama (zidovima) i uskim prolazima (vratima) – tu je mjesto mosnih čvorova. No, ne odvajaju svi uski prolazi sobe. Potrebno je razlučiti koji od skupova $\mathcal{U}_k$, $k = 1, \ldots, U$, predstavljaju uske prolaze koji odvajaju jednu sobu od druge, a koji predstavljaju uske prolaze unutar sobe poput, primjerice, malog razmaka između dviju klupa. Potrebno je naći zatvorenu petlju koja prolazi granicama sobe. Time će se ujedno izdvojiti soba, a i ulazi u sobu, odnosno mosni čvorovi. Valja napomenuti da, ako se radi o prostoriji napučenoj preprekama tako da one formiraju izdvojena područja, onda će ta područja biti proglašena sobama

iako se radi o dijelovima soba. Na taj je način ubrzano hijerarhijsko planiranje, budući da put između dva ograđena dijela prostorije (u ovom slučaju sobe) mora prolaziti uskim prolazom, a taj će put biti inicijalno spremljen kao parcijalni put između mosnih čvorova. Drugim riječima, pozicija mosnog čvora nije nužno na vratima prostorije.

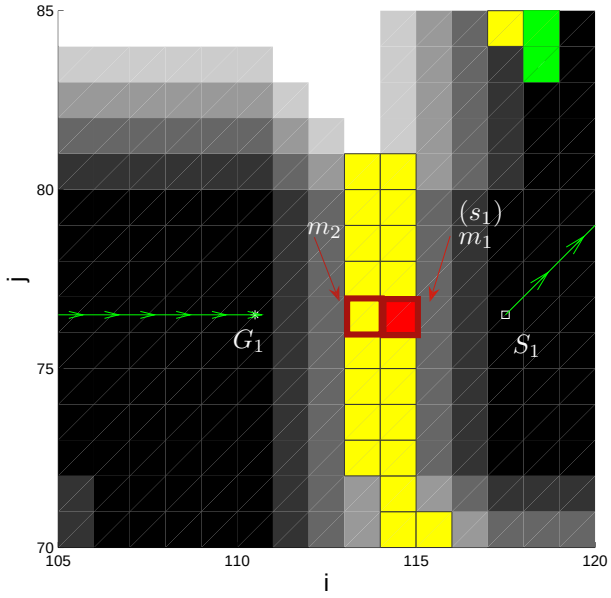
Primijenili smo jednostavnu ideju. Početno je skup mosnih čvorova prazan, $\mathcal{M} \leftarrow \emptyset$. Od svake središnje koordinate uskih prolaza, odnosno koordinate čvora $s_k \in \mathcal{U}_k$, $k = 1, \ldots, U$, odrede se dva najbliža zauzeta polja u suprotnim smjerovima od s_k po osi po kojoj je prisutan lokalni minimum u polju čvora s_k. Jedno polje se proglasi početnim (S_k), a drugo krajnjim (G_k), $k = 1, \ldots, U$, kao na slici 3.10. Stvori se graf



Slika 3.10. *Određivanje pozicije mosnog čvora na središnjoj koordinati uskog prolaza, $s_1 \in \mathcal{U}_1$. Put $\mathcal{P}_1(S_1, G_1)$ zaokružuje sobu. Početne i krajnje pozicije S_k, G_k, $k = 1, \ldots, U$, prikazane su oznakom $\square$, odnosno oznakom $*$.*

$\mathcal{G}_Z(\mathcal{Z}, \mathcal{E}_Z, \mathcal{W}_Z)$ kojemu su čvorovi samo zauzeta polja mrežaste karte zauzeća, a start i cilj su $S_k, G_k \in \mathcal{O}$. Skup bridova $\mathcal{E}_Z$ i pripadajućih težina $\mathcal{W}_Z$ definiran je u skladu s prethodnim definicijama grafa $\mathcal{G}(\mathcal{N}, \mathcal{E}, \mathcal{W})$. Traži se najkraći put $\mathcal{P}_k(S_k, G_k)$ u grafu $\mathcal{G}_Z(\mathcal{Z}, \mathcal{E}_Z, \mathcal{W}_Z)$ od S_k do G_k tako što se od svih drugih S_i i G_i, $i = 1, \ldots, U$, $i \neq k$, napravi zauzeta spojnica preko koje taj put može preći, odnosno čvorovi na spojnici ubace se u graf u skup $\mathcal{Z}$. Promatramo prolaze kao da su ih zauzele neke dinamičke prepreke, a jedino se uski prolaz $\mathcal{U}_k$ smatra slobodnim. Ako postoji put $\mathcal{P}_k(S_k, G_k)$, onda taj put zatvara sobu i s_k postaje mosni čvor čija je pozicija na ulazu u sobu, $\mathcal{M} \leftarrow \mathcal{M} \cup s_k$. Ako postoji paran broj polja u uskom prolazu, gledajući po spojnici između S_k i G_k, da bi se sačuvala optimalnost hijerarhijskog planiranja, postavlja se dodatni mosni čvor do čvora s_k. Tada se koordinate oba mosna čvora nalaze u lokalnom minimumu, a

zajednički brid polja koja opisuju nalazi se na sredini uskog prolaza. Slika 3.10 prikazuje takav slučaj, a njen uvećani dio s naznačenim pozicijama dva mosna čvora jedan do drugog prikazan je na slici 3.11. Ako put ne postoji, onda se ne radi o zatvorenom

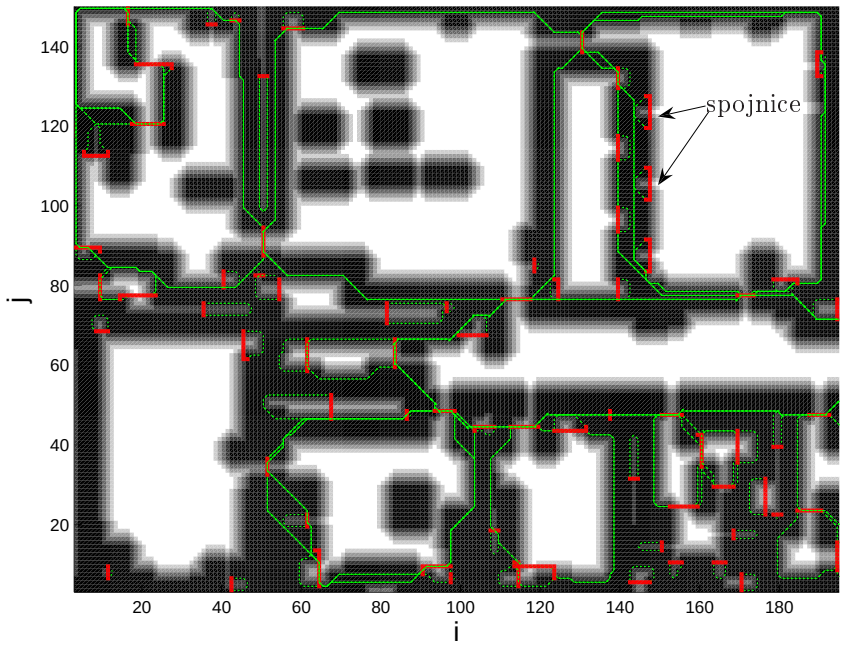


Slika 3.11. *Slučaj parnog broja polja u uskom prolazu: postavljena su dva mosna čvora jedan do drugog* – $m_1, m_2 \in \mathcal{M}$.

području. Središnja koordinata tog uskog prolaza nije ključna za hijerarhijsko planiranje i stoga ona nije pozicija mosnog čvora. Razlog tomu jest što će taj uski prolaz biti unutar neke sobe, a preračunane putanje ne moraju nužno prolaziti kroz njega. Procedura traženja puta $\mathcal{P}_k(S_k, G_k)$ od S_k do G_k i određivanja mosnog čvora, ako put $\mathcal{P}_k(S_k, G_k)$ postoji, ponavlja se za sve $k = 1, \ldots, U$.

3.4.3 Određivanje soba

Cilj ovog (zadnjeg) koraka algoritma jest izdvojiti skup čvorova koji predstavlja polja jedne sobe, a na višoj razini hijerarhije predstavlja jedan potkartni čvor. U prethodnom koraku određeni su ulazi u sobe. Neke sobe neće biti zaokružene putom jer može postojati kraći put od nekog S_k do G_k, $k \in \{1, \ldots, U\}$, koji prolazi sobom manjeg opsega. Međutim, čim je određen put $\mathcal{P}_k(S_k, G_k)$, $k \in \{1, \ldots, U\}$, spojnice između S_k i G_k koje prelaze preko mosnog čvora spremljene su kao granice koje odvajaju jednu sobu od druge i na taj se način lako izdvajaju potkartni čvorovi – sobe. Na slici 3.12 prikazani su određeni putovi kroz granice soba i označene spojnice na ulazima u sobu. Neka je $\mathcal{I} \subset \mathcal{N}$ skup svih čvorova čije se koordinate nalaze na spojnicama između S_k i G_k, za one $k \in \{1, \ldots, U\}$ za koje postoji mosni čvor $s_k \in \mathcal{U}_k$. Neka je $\mathcal{S}$ skup svih čvorova bez spojnica, odnosno $\mathcal{S} \leftarrow \mathcal{N} \setminus \mathcal{I}$. Skup $\mathcal{S}$ razdjeljuje se na S disjunktnih podskupova. Svaki podskup $\mathcal{S}_i$, $i = 1, \ldots, S$, opisuje jednu sobu. Svaki element skupa $\mathcal{S}$ raspoređuje



Slika 3.12. *Granice koje izdvajaju sobe – putovi i spojnice.*

se u odgovarajući podskup $\mathcal{S}_i$. Početno je $\mathcal{S}_S \leftarrow \{n\} \in \mathcal{S}$, $\mathcal{S} \leftarrow \mathcal{S} \setminus \{n\}$ i $S = 1$, a dalje se iterativno rješava sljedeći izraz:

$$\begin{aligned} &\text{ako } \exists m \in \mathcal{S} \text{ i } n \in \mathcal{S}_S \text{ tako da } \{m,n\} \in \mathcal{E} \\ &\text{tada } \mathcal{S}_S \leftarrow \mathcal{S}_S \cup \{m\} \text{ i } \mathcal{S} \leftarrow \mathcal{S} \setminus \{m\} \\ &\text{inače } S \leftarrow S+1, \quad \mathcal{S}_S \leftarrow \{n\} \in \mathcal{S}, \quad \mathcal{S} \leftarrow \mathcal{S} \setminus \{n\}. \end{aligned} \tag{3.20}$$

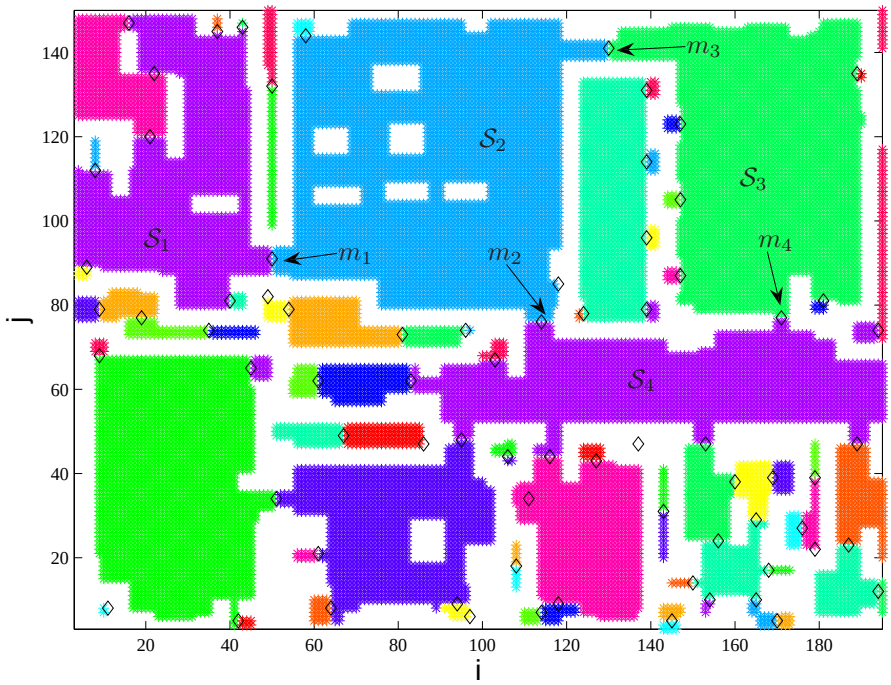
Postupak (3.20) ponavlja se sve dok nije $\mathcal{S} = \emptyset$. Naposljetku se spojnice pridjeljuju proizvoljno sobama tako da vrijedi:

$$\begin{aligned} &\text{za } \forall n \in \mathcal{I} \\ &\text{ako je } m \in \mathcal{S}_i, \text{ za } i \in \{1,\ldots,S\}, \text{ tako da } \{m,n\} \in \mathcal{E} \\ &\text{tada } \mathcal{S}_i \leftarrow \mathcal{S}_i \cup \{n\}. \end{aligned} \tag{3.21}$$

Mosni se čvorovi pridjeljuju dvjema sobama. Tako će jedan mosni čvor, koji je na spojnici između dviju soba, biti pridjeljen i jednoj i drugoj sobi. Prema tome za mosne čvorove vrijedi:

$$\begin{aligned} &\text{za } \forall m \in \mathcal{M} \\ &\text{ako je } n \in \mathcal{S}_i \text{ i } l \in \mathcal{S}_j \text{ za } i,j \in \{1,\ldots,S\} \text{ i } i \neq j, \text{ tako da } \{m,n\},\{l,m\} \in \mathcal{E} \\ &\text{tada } \mathcal{S}_i \leftarrow \mathcal{S}_i \cup \{m\} \text{ i } \mathcal{S}_j \leftarrow \mathcal{S}_j \cup \{m\}. \end{aligned} \tag{3.22}$$

Na slici 3.13 prikazane su sobe kao skupovi čvorova iste boje i mosni čvorovi označeni s ◊. Radi jasnoće, samo su četiri sobe numerirane. Prostor prikazan na slici 3.13 odgovara prostoru prikazanom na slici 3.12.



Slika 3.13. *Opis dijela prostora sobama i mosnim čvorovima (oznaka ◊).*

3.4.4 Organiziranje viših razina hijerarhijske apstrakcije

Na razini R_D je graf stvoren iz mrežaste karte zauzeća, a na razini R_{D-1} je graf sastavljen od mosnih čvorova i soba. Mosni čvorovi povezuju potkartni čvor u kojem se nalaze s potkartnim čvorom više razine i to tako da se dupliciraju u razini iznad. Pri tome su mosni čvorovi u razini R_D određeni pozicijom, a u razini R_{D-1} prikazani apstraktno. Veze između mosnih čvorova u razini R_{D-1} su parcijalni putovi određeni u nižoj razini hijerarhijske apstrakcije. Parcijalni putovi određuju se na razini R_D između svih mosnih čvorova koji pripadaju istom potkartnom čvoru, odnosno sobi.

Opisana hijerarhijska karta ima ukupno tri razine hijerarhijske apstrakcije ($D = 2$), a najapstraktnija razina R_0 sadrži samo jedan čvor. Može imati i četiri razine ako se iz skupa $\mathcal{M}$ ručno izdvoje mosni čvorovi koji se nalaze na ulazima u dizala, a u razini R_1 spajaju potkartne čvorove katove. Međutim, moguće je primijeniti jednostavan proces grupiranja čvorova i stvoriti dodatne apstraktnije razine.

Neka se grupira $n \in \mathbb{N}$ potkartnih čvorova (soba) i pripadnih mosnih čvorova tako da oni čine jedan potkartni čvor na višoj razini apstrakcije. Da bi se sačuvala konzistentnost hijerarhije, potrebno je grupirati bliske potkartne čvorove i pripadne mosne čvorove. Dva

su potkartna čvora bliska ako vrijedi:

$$\begin{aligned}&\exists m \in \mathcal{M} \text{ i } i, j \in \{1, \ldots, S\},\ i \neq j, \text{ tako da } m \in \mathcal{S}_i \text{ i } m \in \mathcal{S}_j \\ &\text{tada su } \mathcal{S}_i \text{ i } \mathcal{S}_j \text{ bliski.}\end{aligned} \tag{3.23}$$

Skupom povezanih potkartnih čvorova zovemo takav skup $\mathcal{X}$ u kojem postoji barem $|\mathcal{X}| - 1$ bliskih parova potkartnih čvorova.

Neka je $\mathcal{K}_j^{D-2}$ skup s n povezanih potkartnih čvorova, $1 < n < S$, i pripadnih mosnih čvorova i neka ima $K^{D-2} \in \mathbb{N}$ takvih skupova, odnosno $j \in \{1, \ldots, K^{D-2}\}$. Skupovi $\mathcal{K}_j^{D-2}$, $j = 1, \ldots, K^{D-2}$, su potkartni čvorovi na razini R_{D-2}. Unutar svakog skupa $\mathcal{K}_j^{D-2}$, $j = 1, \ldots, K^{D-2}$, postoji barem jedan mosni čvor $r \in \mathcal{M}$ koji se nalazi u samo jednoj sobi $r \in \mathcal{S}_i \in \mathcal{K}_j^{D-2}$, $i \in \{1, \ldots, S\}$ i u nijednoj drugoj sobi iz $\mathcal{K}_j^{D-2}$, odnosno $r \notin \mathcal{K}_j \setminus \mathcal{S}_i$. Takav je čvor rubni mosni čvor (prema definiciji u odlomku 3.3.2). Rubni mosni čvor $r \in \mathcal{M}$ koji se nalazi u sobi $\mathcal{S}_i \in \mathcal{K}_j^{D-2}$, $j \in \{1, \ldots, K^{D-2}\}$, $i \in \{1, \ldots, S\}$, postoji kao rubni mosni čvor u sobi $\mathcal{S}_k \in \mathcal{K}_l^{D-2}$, $l \in \{1, \ldots, K^{D-2}\}$, $l \neq j$, $k \in \{1, \ldots, S\}$, koja je bliska sobi $\mathcal{S}_i$. Prema tome, rubni mosni čvorovi povezuju druge potkartne čvorove na razini R_{D-2}, kao što svi mosni čvorovi određeni u razini R_D povezuju potkartne čvorove - sobe na razini R_{D-1}. Rubni mosni čvorovi iz razine R_{D-1} nalaze se kao mosni čvorovi u razini R_{D-2}. Veze između mosnih čvorova na razini R_{D-2} su parcijalni putovi određeni u nižoj razini R_{D-1} između rubnih mosnih čvorova. Parcijalni putovi određuju se između svih rubnih mosnih čvorova koji pripadaju istom potkartnom čvoru $\mathcal{K}_j^{D-2}$, $j \in \{1, \ldots, K^{D-2}\}$. Zapravo su rubni mosni čvorovi pravi mosni čvorovi po prvotnoj definiciji danoj u odlomku 3.1.2 jer povezuju potkartni čvor u kojem se nalaze s potkartnim čvorom više razine. Svi su ostali mosni čvorovi unutar $\mathcal{K}_j^{D-2}$ unutarnji čvorovi i ne koriste se za računanje parcijalnih putova između svakog od njih, jer su nužni putovi određeni u razini ispod. Za objašnjenje razmotrimo razinu R_D. Budući da jedan mosni čvor pripada dvjema sobama, bit će povezan parcijalnim putom sa svim mosnim čvorovima iz te dvije sobe. Drugim riječima, svake dvije sobe koje dijele mosni čvor bit će povezane parcijalnim putovima. Stoga na razini R_{D-1} nije potrebno računati parcijalne putanje između svih mosnih čvorova unutar potkartnog čvora $\mathcal{K}_j^{D-2}$, $j \in \{1, \ldots, K^{D-2}\}$ jer se putanje između dva mosna čvora koja nisu u bliskim sobama jednostavno odrede unijom odgovarajućih parcijalnih putanja, za što se brine algoritam pretraživanja mosnih čvorova u toj razini (odlomak 3.3.3). Parcijalne su putanje potrebne između rubnih mosnih čvorova unutar tog potkartnog čvora zbog pretraživanja čvorova u višoj razini hijerarhijske apstrakcije.

Postupak grupiranja povezanih potkartnih čvorova može se nastaviti dalje dodavanjem apstraktnije hijerarhijske razine, dok je god moguće raditi grupiranje, odnosno dok se ne stvori razina u kojoj postoje samo dva potkartna čvora. Te potkartne čvorove označavamo s $\mathcal{K}_k^{D-x}$, gdje je $k = 1, \ldots, K^{D-x}$ indeks potkartnog čvora u razini R_{D-x}, a $x = 3, \ldots, D$ odmak od najniže razine hijerarhijske apstrakcije. Dodavanjem apstraktnijih razina dimenzija hijerarhije raste, odnosno svakom novom razinom postaje $D \leftarrow D + 1$. Razlog označavanju potkartnih čvorova koji se odnose na sobe sa $\mathcal{S}_i$, a

potkartnih čvorova na višim razinama sa $\mathcal{K}_j^{D-x}$, leži u tome što su sobe određene iz stvarne karte prostora i one čine osnovu H-grafa, a svi ostali potkartni čvorovi na višim razinama određeni su iz apstraktnih čvorova H-grafa.

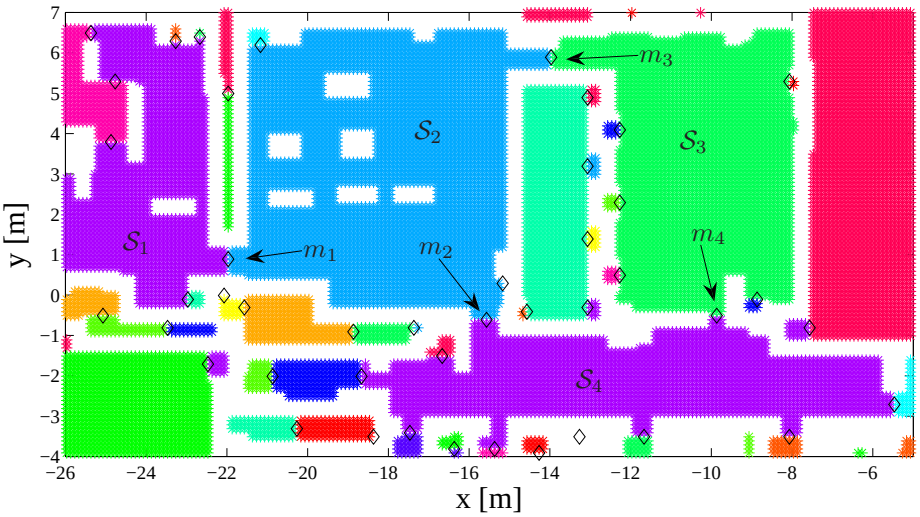
Na slici 3.13 postoje nepovezane sobe s ostalim sobama, odnosno zaokružena područja zidovima sa svih strana. Takve sobe ispuštaju se iz hijerarhijske karte jer nisu potrebne u hijerarhijskom planiranju putanje.

Neki mosni čvorovi mogu biti rubni na više razina, ovisno o složenoj hijerarhijskoj apstrakciji prostora. Primjerice, mosni čvor u sobi $\mathcal{S}_i$, $i \in \{1, \ldots, S\}$, na razini R_D, nalazi se kao rubni mosni čvor u potkartnom čvoru $\mathcal{K}_j^{D-2}$, $j \in \{1, \ldots, K^{D-2}\}$ na razini R_{D-1} i kao rubni mosni čvor unutar potkartnog čvora $\mathcal{K}_l^{D-3}$, $l \in \{1, \ldots, K^{D-3}\}$. Vertikalni mosni čvorovi moraju biti odabrani iz skupa $\mathcal{M}$ ručno. Smješteni su na vratima dizala. Oni se pojavljuju kao rubni mosni čvorovi u razinama R_D, $R_{D-1}, \ldots, R_{i+1}$, gdje razina R_i, $i < D$, sadrži potkartne čvorove koji predstavljaju katove. Na toj razini su prikazane veze katova vertikalnim mosnim čvorovima.

3.5 Eksperimentalni rezultati

Predloženi algoritam FHD* testiran je na karti Zavoda. Usporedili smo ga s algoritmima D*, FD* i HD* pod istim uvjetima [84]. Izgrađena je mrežasta karta zauzeća i organizirane su u tri hijerarhijske razine, R_0, R_1 i R_2. Najniža razina R_2 sadrži graf stvoren iz mrežaste karte zauzeća i opisuje detalje prostora. Mobilni robot senzorima otkriva dinamičke prepreke i osvježava informaciju o zauzetosti u mrežastoj karti.

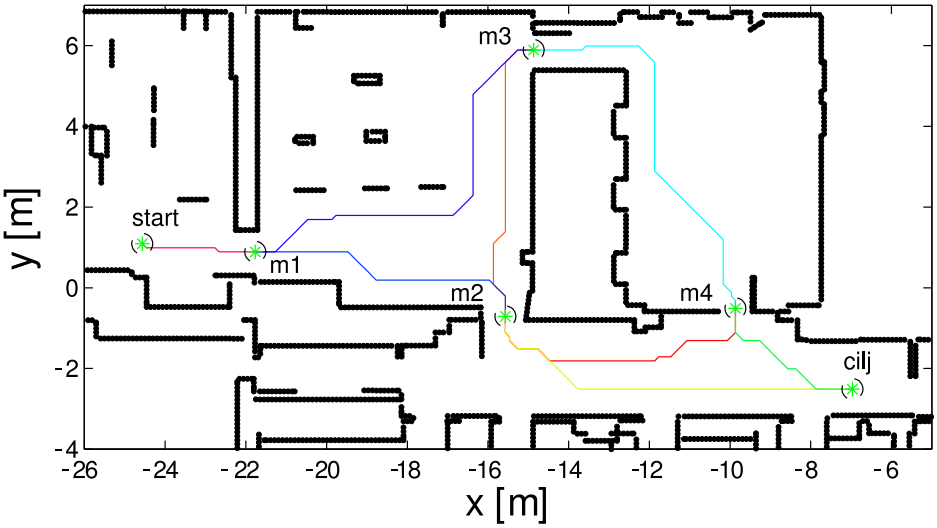
U razini R_1 nalaze se potkartni čvorovi koji predstavljaju sobe i hodnike Zavoda. Slika 3.14 prikazuje dio Zavoda na kojem su se izvodili eksperimenti. Prikazane su sobe



Slika 3.14. *Prikaz potkartnih čvorova i mosnih čvorova na razini R_1 u dijelu prostora u kojem su se izvodili eksperimenti.*

određene grupiranjem čvorova u razini R_2 metodom opisanom u odlomku 3.4. Označene su samo sobe i mosni čvorovi koji su bili korišteni u eksperimentu. Razina R_0 sadrži samo jedan čvor – kat. Algoritam dinamičkog prozora opisan u 5. poglavlju korišten je za slijeđenje putanje.

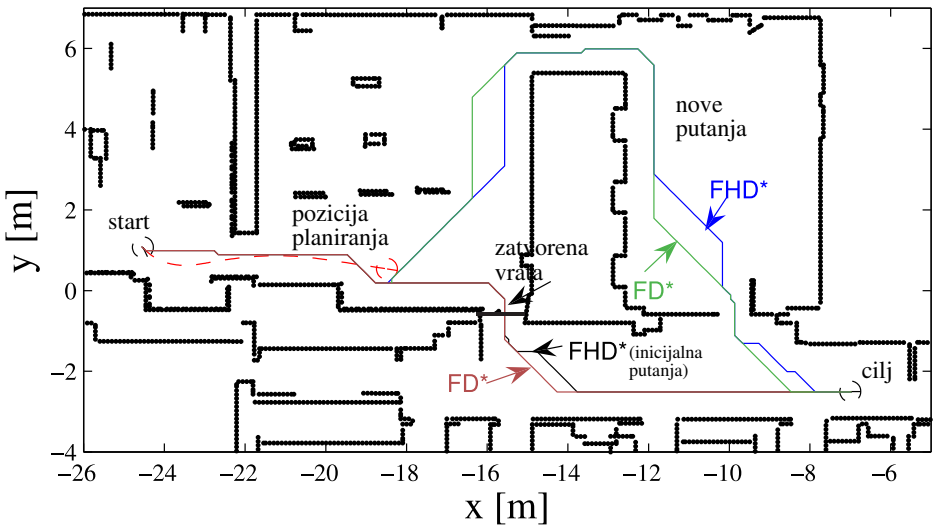
Na slici 3.15 prikazan je skup parcijalnih putanja korišten u hijerarhijskom planiranju putanje od starta do cilja. Start se nalazi u potkartnom čvoru $\mathcal{S}_1$, a cilj se nalazi u potkartnom čvoru $\mathcal{S}_4$. Ovaj skup parcijalnih putanja koriste oba hijerarhijska algoritma, HD* i FHD*. Prema tome, prednost postavljanja mosnih čvorova na optimalna mjesta nije naglašena u eksperimentu i smatra se da HD* radi s istom hijerarhijskom kartom kao FHD*. Inicijalno pretraživanje je jednako za oba hijerarhijska algoritma.



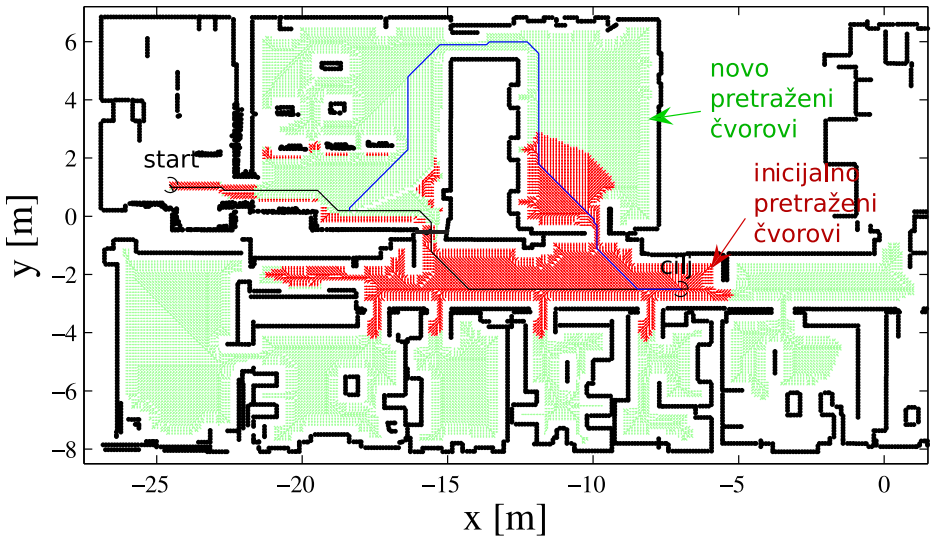
Slika 3.15. *Skup parcijalnih putanja između mosnih čvorova.*

Eksperimenti su osmišljeni tako da hijerarhijski algoritam mora pretražiti sve razine hijerarhijske apstrakcije kad robot otkrije nepoznatu prepreku na inicijalnoj putanji. Nehijerarhijski algoritmi moraju pretražiti gotovo cijeli graf. Na slici 3.16 uspoređeni su inicijalne i dinamički isplanirane putanje za algoritme FHD* i FD*. Putanje su optimalne, ali geometrijski različite zbog postojanja više optimalnih putanja, a korištenje heuristike prema različitim startovima uzrokuje odabir različitih putanja. Kad robot stigne do pozicije označene kao "pozicija dinamičkog planiranja" (slika 3.16), otkrije laserskim senzorom udaljenosti da su vrata zatvorena. Odvožena trajektorija do te pozicije označena je crtkanom linijom. Usporedba putanja algoritama HD* i D* nije potrebna jer se dobiju slični rezultati. Sljedeće slike prikazuju pretraženo područje za svaki od algoritama: FD*, D*, HD* i FHD*. Redoslijed slika odabran je prema broju pretraženih čvorova.

Na slici 3.17 prikazano je pretraženo područje algoritmom FD*. Unutrašnje naznačeno područje pretraženo je inicijalnim izračunom, a vanjsko je pretraženo u procesu dinamičkog planiranja putanje zbog otkrivene nove prepreke. Budući da algoritam FD*

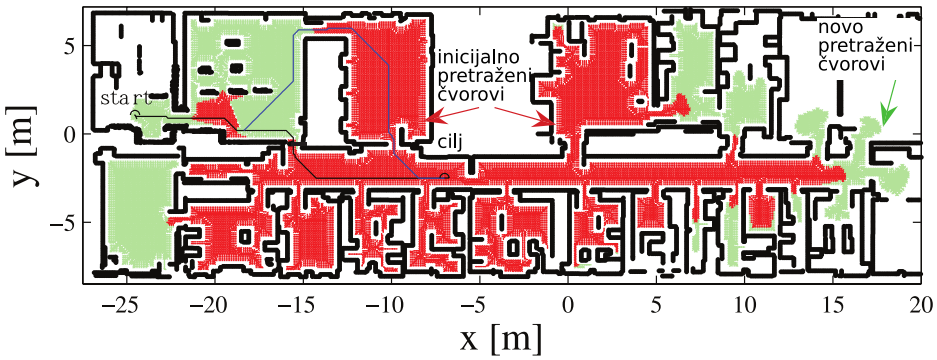


Slika 3.16. *Usporedba inicijalne i dinamički isplanirane putanje određene algoritmima FHD* i FD*.*

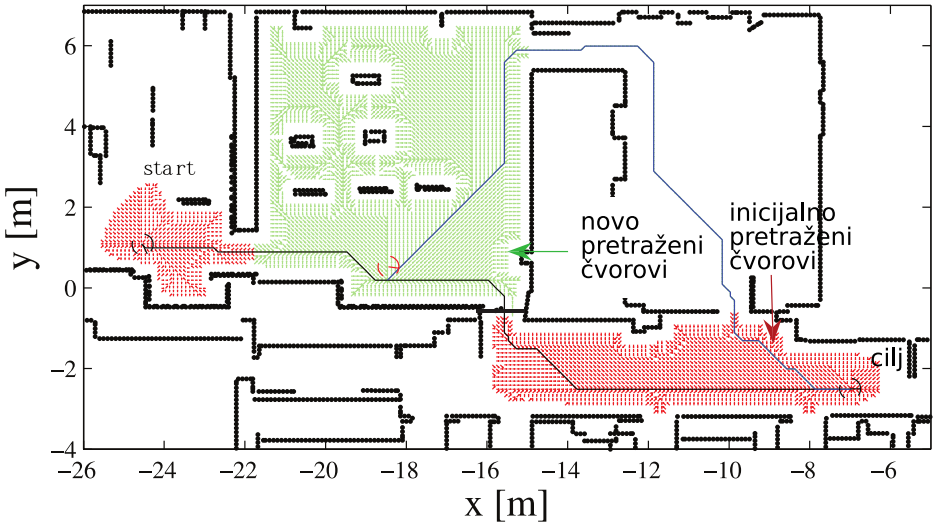


Slika 3.17. *Pretraženo područje algoritmom FD*.*

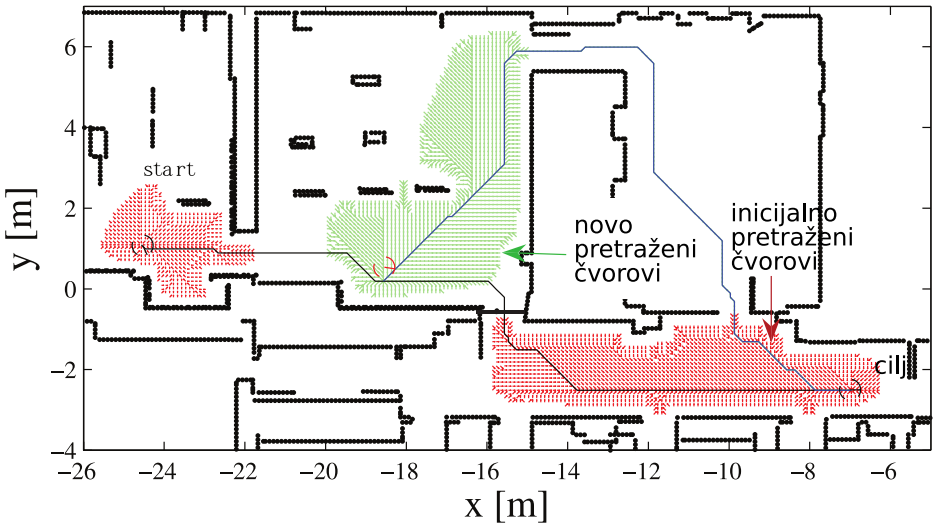
koristi heuristiku, minimalno područje pretraženo je inicijalnim izračunom. Proces dinamičkog planiranja nastavlja od ruba područja, koje je određeno inicijalnim planiranjem. Područje se širi sve dok se ne odredi nova putanja. Na slici 3.18 prikazano je pretraženo područje algoritmom D*. Budući da algoritam D* ne koristi heuristiku, područje pretraženo inicijalnim izračunom širi se kružno oko cilja. Proces dinamičkog planiranja nastavlja od ruba pretraženog područja inicijalnog izračuna, koje je mnogo šire od pretraženog područja algoritmom FD* u inicijalnom izračunu. Zato je algoritmu D* potrebno manje vremena za određivanje nove putanje. Na slici 3.19 prikazano je pretraženo područje algoritmom HD*. Područje pretraženo inicijalnim izračunom određeno



Slika 3.18. *Pretraženo područje algoritmom D*.*



Slika 3.19. *Pretraženo područje algoritmom HD*.*



Slika 3.20. *Pretraženo područje algoritmom FHD*.*

je pozicijom starta u odnosu na mosne čvorove iz potkartnog čvora u kojem se nalazi start i pozicijom cilja u odnosu na mosne čvorove iz potkartnog čvora u kojem se nalazi cilj. Od cilja se moraju odrediti sve putanje do mosnih čvorova iz istog potkartnog čvora u kojem se nalazi cilj. Isto tako se i od starta moraju odrediti putanje do svih mosnih čvorova iz potkartnog čvora u kojem se nalazi start. Dalje se pretražuju samo mosni čvorovi. U procesu dinamičkog planiranja, pretražuju se samo čvorovi iz potkartnog čvora u kojem se nalazi pozicija dinamičkog planiranja i mosni čvorovi na višoj razini. Na slici 3.20 prikazano je pretraženo područje algoritmom FHD*. Pretraženo je područje u inicijalnom izračunu jednako kod oba hijerarhijska algoritma, ali je zato pretraživanje u procesu dinamičkog planiranja znatno smanjeno algoritmom FHD*.

Tablica 3.1. *Usporedba algoritama planiranja putanje u inicijalnom procesu*

Alg.	duljina putanje [polja]	broj pretraženih čvorova	broj iteracija	vrijeme planiranja putanje[ms]
D*	187	27360	21063	272 ms
FD*	187	6767	4783	36 ms
HD*	187	3150	2082	25 ms
FHD*	187	3159	2082	25 ms

Tablica 3.2. *Usporedba algoritama planiranja putanje u procesu dinamičkog planiranja*

Alg.	duljina putanje [polja]	broj pretraženih čvorova	broj iteracija	vrijeme dinamičkog planiranja [ms]
D*	188	14518	11395	120 ms
FD*	188	17796	12358	130 ms
HD*	188	5822	5571	70 ms
FHD*	188	3398	2945	41 ms

Rezultati usporedbe algoritama prikazani su u tablici 3.1 za izračune inicijalne putanje i u tablici 3.2 za izračune dinamičkog planiranja putanje. Duljina putanje, koja je usko vezana s cijenom putanje, jednaka je za sve algoritme. Broj pretraženih čvorova ključni je pokazatelj računske složenosti algoritama. Broj iteracija odnosi se na ponavljanje izvođenja glavne *dok* petlje algoritama pretraživanja.

3.6 Sažetak

U ovom je poglavlju opisan predloženi algoritam FHD* koji osigurava optimalnost putanje i poboljšava dinamičke karakteristike Cagigasovog algoritma HD* [18]. Značajno su smanjeni vrijeme izračuna, cijena putanje i utrošena memorija. Predložena je strategija optimalnog postavljanja mosnih čvorova – ključnih točaka između kojih su izračunate parcijalne putanje. Ubrzan je proces dinamičkog planiranja korištenjem heuristike prema trenutnoj poziciji robota, čime je dinamičko planiranje fokusirano oko optimalne putanje, koja je određena pretraživanjem minimalnog broja čvorova. Izvedena je nova hijerarhijska organizacija karte kojom je osigurano optimalno horizontalno i vertikalno planiranje putanje. Prikazana je metoda automatske izgradnje takve hijerarhijske karte iz mrežaste karte zauzeća. Metoda jamči konzistentnost stvorene hijerarhije i omogućuje optimalno hijerarhijsko planiranje putanje u dinamičkim prostorima. Uspješnost algoritma FHD* usko je povezana s hijerarhijskom kartom, koja koristi znatno manje materijaliziranih cijena – parcijalnih putanja između mosnih čvorova i time štedi memoriju. Eksperimenti su potvrdili očekivane prednosti predloženog algoritma FHD*.

Poglavlje 4

Planiranje putanje u nepoznatim unutarnjim prostorima

U prethodna dva poglavlja opisani su algoritmi planiranja putanje mobilnog robota u prostorima kojima robot poznaje kartu te algoritmi dinamičkog planiranja putanje kada se u prostoru pojave prepreke koje nisu modelirane u karti. U ovome se poglavlju opisuju planiranja putanje mobilnog robota u njemu potpuno nepoznatim prostorima, tj. u prostorima u kojima robot ne poznaje kartu. Zadaća je robota autonomno se gibati kroz takav prostor i izgraditi kartu. Pri tome se zahtijeva da robot istraži čitavi prostor pa se problem planiranja putanje u nepoznatim prostorima obično naziva istraživanjem prostora. Dodatni je zahtjev da robot pritom prijeđe što je moguće manju udaljenost.

Strategija istraživanja prostora obično pretpostavlja da je mobilni robot točka u neistraženom prostoru u kojem se nalazi konačan broj prepreka postavljenih nasumice, različitih oblika i veličina. Problem autonomnog istraživanja prostora razmatra se odvojeno od problema istovremene lokalizacije robota u tom prostoru, tj. pretpostavlja se da je riješena lokalizacija.

Ovo je poglavlje strukturirano kako slijedi. U potpoglavlju 4.1 dan je pregled strategija istraživanja nepoznatog prostora. Detaljnije su opisane dvije strategije: strategija promatrača i Ekmanova strategija. Unutarnje je prostore najjednostavnije predstaviti skupom povezanih linija, tj. poligonom. Tako je u potpoglavlju 4.2 opisan algoritam izdvajanja linijskih segmenata iz laserskog senzora udaljenosti. Potpoglavlje 4.3 opisuje izvedeni algoritam istraživanja poligonalnih prostora koji objedinjuje utjecaj nesigurnosti senzorskih očitanja i položaja mobilnog robota i postiže konvergenciju u realnim uvjetima. Naposljetku, eksperimentalni rezultati na stvarnom robotu, prikazani su u potpoglavlju 4.4.

4.1 Pregled strategija istraživanja nepoznatih prostora

Radovi Brooksa [14] i Oommena [88] među prvima su u tom području. U njihovim radovima se ovaj problem naziva prikupljanjem informacija o terenu (engl. *terrain acquisition*).

Većina algoritama koji rješavaju problem istraživanja prostora radi s pretpostavkom na određeni oblik prepreka. U [88] se pretpostavljaju konveksne poligonalne prepreke. U tom se radu koristi strategija u kojoj je robot sposoban obaviti dvije elementarne akcije: skeniranje, da bi identificirao sve vidljive vrhove prepreka sa svoje trenutne pozicije, i kretanje tj. pomicanje robota po pravocrtnoj liniji. Robot se kreće od vrha do vrha prepreka. Da bi robot istražio okolinu sa n vrhova prepreka, algoritmu je potrebno n operacija skeniranja i najviše $2(n-1)$ pomicanja između vrhova.

Strategija promatrača [76] zasniva se na okruživanju prepreka proizvoljnog oblika. Strategija se može poistovjetiti s algoritmom praćenja zida. Strategija prema Matarić [78] zasniva se na algoritmu praćenja zida u kombinaciji s izbjegavanjem prepreka i topološkim planiranjem putanje. Međutim, strategija nije učinkovita u složenijim unutarnjim prostorima.

Postoji nekoliko strategija istraživanja koje guraju robota na granicu istraženog i neistraženog prostora. Strategija prema radu Ekmana *et al.* [27] zasniva se na postojanju diskontinuiteta u podacima koje daje senzor udaljenosti. Ti se diskontinuiteti nazivaju skočnim bridovima (engl. *jump edges*). Skočni bridovi dijele istražena i neistražena područja. Prema tome, ako robot obavi mjerenje ispred skočnog brida, otkrit će se novo područje u prostoru. Prostor je istražen kada u karti više ne postoji niti jedan skočni brid. Pod određenim uvjetima moguće je stvoriti vjernu kartu prostora uz osiguranu konvergenciju algoritma. Informacije o prostoru sadržane su u dvije apstraktne razine. Za prikaz prostora se na donjoj razini koriste poligoni, a na gornjoj razini grafovi. Poligonalni prikaz određuje kartu, a prikaz se grafom koristi za planiranje putanje. Algoritam istraživanja zanemaruje nesigurnosti mjerenja senzora i pozicije robota.

Strategija prema [131] zasniva se također na diskontinuitetima u podacima senzora, koje nazivaju otvorima. Pretpostavljaju da je prostor poligonalnog oblika. Konstruiraju posebnu dinamičku strukturu podataka nazvanu navigacijskim stablom otvora (engl. *gap navigation trees*). Ideja je minimizirati potrebne informacije koje robot mora sakupiti da bi riješio neki zadatak. Koriste tzv. senzor otvora, koji se može konstruirati iz senzora udaljenosti i smatraju da robot ne koristi niti jedan drugi senzor osim senzora otvora. Sve akcije robota svedene su na "lovljenje otvora". Izgled prostora u odnosu na poziciju robota kodira se kao stablo koje označava kritične događaje – otvori pri gibanju robota mogu nestati, pojaviti se, prelomiti i spojiti. Analizirajući način na koji se kritični događaji pojavljuju u senzoru otvora, izgrađuje se stablasta reprezentacija koja označuje optimalno gibanje u prostoru, iako se precizna mjerenja ne mogu napraviti. Strategija ne zahtijeva egzaktnu lokalizaciju i potpunu izgradnju karte, što je njezina važna prednost. Međutim, neki praktični aspekti ostali su neriješeni, kao što su zašumljena očitanja senzora otvora. Robot u tom slučaju može loviti krivi otvor.

Yamauchi *et al.* [138] svoju strategiju nazivaju istraživanjem granica (engl. *frontier based exploration*). Njihov algoritam ne radi nikakve pretpostavke na oblik prepreka i primjenjiv je u realnim uvjetima. Algoritam koristi mrežastu kartu zauzeća za pohranjivanje otkrivenih informacija iz prostora, iz koje pomoću tehnika računalnog vida izdvaja granice između slobodnog prostora i nepoznatog prostora. Strategija određuje najbližu točku granice kao sljedeću poziciju koju robot mora zauzeti. Nedostatak algoritma je velika računska složenost jer otkrivanje granica i odabiranje najobećavajućije granice zahtijeva veliki memorijski prostor i složen izračun.

Strategija prema [4] koristi točke zaklanjanja koje predstavljaju granice između otkrivenog i neotkrivenog prostora. Strategija pretpostavlja senzor s beskonačnim dometom i poligonalne prepreke. Strategija se zasniva na spremanju i obilaženju samo točaka zaklanjanja. Izbjegnuto je računanje granica istraženog i neistraženog prostora i spremanje cijele karte, čime se ubrzava izračun. Međutim, strategija ima problem lokalnog minimuma u slučaju kada je robot okružen preprekama i nema niti jedne neobiđene točke zaklanjanja u svojoj okolini. Budući da algoritam ne upisuje prepreke u kartu, nego samo točke zaklanjanja, ne može planirati putanja do točaka koje trenutno nisu vidljive te je prema tome ova metoda lokalna. Slična je strategija i "poboljšanje pogleda" [39] koja se također zasniva na zaklonjenim točkama i pretpostavlja idealnu lokalizaciju i idealna mjerenja.

Sve spomenute metode većinom nisu pouzdane kada se primjene u realnim uvjetima zbog nesigurnosti mjerenja, lokalizacije i izgradnje karte.

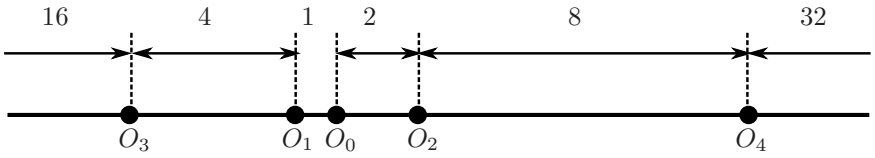
U nastavku su pobliže opisane neke strategije za istraživanje nepoznatog prostora koje se zasnivaju na radovima Lumelskyog *et al.* [76] i strategija Ekmana *et al.* [27] na kojoj se zasniva strategija predložena u ovom radu.

4.1.1 Strategija promatrača

Strategija promatrača pretpostavlja konačan ili beskonačan prostor koji se istražuje, uz uvjet da su prepreke međusobno vidljive, tj. bilo koji par prepreka povezan je preko niza prepreka koje su jedna drugoj vidljive. U početnoj poziciji robot otkriva sve vidljive bridove prepreka. Ako nema prepreka, zadatak robota je završen, a inače se robot pravocrtno giba prema najbližoj prepreci te se zatim počinje gibati oko nje (okružuje ju) te usput nadopunjuje kartu. Robot zatim obilježava prepreku kao posjećenu, pronalazi na karti sljedeću neposjećenu prepreku te se počinje gibati prema njoj. Ova se procedura ponavlja sve dok ne nestane neposjećenih prepreka. Ako se tijekom gibanja od jedne do druge prepreke otkrije neka nova prepreka, robot prvo okruži tu novootkrivenu prepreku te zatim odabire novu neposjećenu prepreku prema kojoj se počinje gibati. Ako pak robot usred gibanja naiđe na posjećenu prepreku, jednostavno je zaobiđe najkraćom putanjom i nastavlja svoje pravocrtno gibanje. Kako je geometrijski oblik te posjećene prepreke već poznat, pronalaženje kraće putanje ne bi trebalo predstavljati problem.

U svakoj akciji pravocrtnog gibanja jedna neposjećena prepreka dana je kao novi cilj pretrage i najmanje je jedna neposjećena prepreka zapravo istražena. Kako pros-

tor koji se istražuje ima konačan broj prepreka konačnih dimenzija i kako niti jedna posjećena prepreka ne može postati novi cilj za pravocrtno gibanje, ova procedura ima zajamčenu konvergenciju. Medutim, kako će prikazati sljedeći primjer, gornja granica duljine putanje ove jednostavne strategije nema toliko dobru učinkovitost. Pretpostavimo jednodimenzionalan slučaj sa n prepreka, kao što je prikazano na slici 4.1. Radi



Slika 4.1. *Slučaj u kojem će opisana procedura imati kvadratičnu složenost izvođenja* $O(n^2)$.

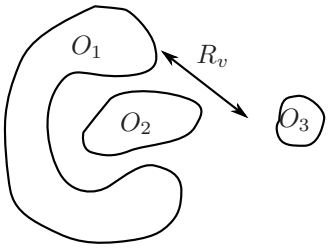
jednostavnosti, svaka je prepreka, $O_0, O_1, \ldots, O_n$, predstavljena kao točka. Neka su prepreke s neparnim indeksima pozicionirane lijevo od O_0, a one s parnim indeksima desno od O_0. Neka je O_1 udaljena $2^0 = 1$ jedinica od O_0, neka je O_2 udaljena $2^1 = 2$ jedinica od O_0, neka je O_3 udaljena $2^2 = 4$ jedinica od O_1, neka je O_4 udaljena $2^3 = 8$ jedinica od O_2, itd., i na kraju, neka je O_{n-1} udaljena 2^{n-2} jedinica od O_{n-3}. Neka je polumjer kruga vidljivosti oko robota R_v. Pretpostavimo da je $R_v > 2^{n-2}$ tako da je svaki par susjednih prepreka vidljiv jedan drugome.

Počevši od O_0, robot može vidjeti i O_1 i O_2; krene prema O_1 jer je bliže. U O_1, vidi novu prepreku O_3, ali kako je sada O_2 najbliža neposjećena prepreka, robot krene prvo prema O_2. Nastavljajući po ovoj proceduri sve prepreke će na kraju biti posjećene numeričkim redoslijedom po njihovim indeksima. Prateći putanju može se opaziti da je O_0 prepreka bila prijeđena (n-1) puta, O_1 je bila prijeđena (n-2) puta itd. Očito je najgori slučaj izvođenja algoritma kvadratične složenosti po preprekama, dakle kvadratična je složenost i po opsezima prepreka i po udaljenostima između prepreka.

Kasnije će biti pokazano da određene promjene u proceduri poboljšavaju izvođenje algoritma iz kvadratične u linearnu složenost. Nakon što robot zaokruži prepreku i zabilježi da je posjećena, robot će izabrati između svih vidljivih prepreka (za razliku od svih neposjećenih prepreka) onu najbližu neposjećenu prepreku i sljedeću će ju istražiti; ako nijedna takva prepreka ne postoji, robot će se vratiti na zadnju posjećenu prepreku i od tamo tražiti najbližu posjećenu vidljivu prepreku. Kako je sljedeća prepreka za pretragu vidljiva s trenutne prepreke, robot može uvijek pronaći putanju prema njoj bez susretanja drugih prepreka. Jedino će posjetiti prethodno posjećenu prepreku prilikom vraćanja. Cijeli se proces ponavlja dok postoji barem jedna neposjećena prepreka.

Može se dogoditi da niti jedna od trenutno identificiranih neposjećenih prepreka ne bude vidljiva s trenutne pozicije robota. Koristeći u međuvremenu izgrađenu kartu robot odabire koju će sljedeću prepreku posjetiti (primjerice najbližu). Postavlja se pitanje kojom bi putanjom robot trebao doći do prepreke. Odgovor opet daje vraćanje na prethodno posjećenu prepreku. Vraćanje jamči da generirana putanja predstavlja

razgranato stablo, koje pojednostavljuje algoritam. Nadalje, ravne linije segmenata putanje između prepreka ograničene su polumjerom vidljivosti R_v.



Slika 4.2. *Prepreka O_3 jedino se može vidjeti s prepreke O_2. Iako je O_2 potpuno poznata nakon istraživanja O_1, robot svejedno mora okružiti O_2 ne bi li propustio istražiti prepreku O_3.*

Treba imati na umu da prepreka koja nije bila posjećena, ali je potpuno pretražena ne može biti zabilježena kao posjećena. Svaka neposjećena prepreka mora biti eksplicitno posjećena jer postoji mogućnost da postoje druge prepreke koje su vidljive samo s određene prepreke. To je prikazano na slici 4.2. U procesu gibanja oko prepreke O_1 robot će potpuno pretražiti i prepreku O_2. Ipak, prepreka O_2 mora bit eksplicitno posjećena jer inače robot ne bi nikad otkrio prepreku O_3 uz zadani polumjer vidljivosti R_v.

4.1.2 Ekmanova strategija

Pojmovi poligona

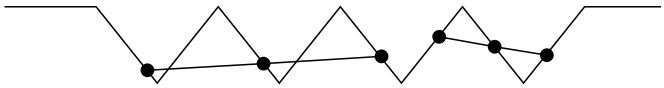
Poligon je dio ravnine omeđen s najmanje tri dužine $\overline{v_1v_2}, \overline{v_2v_3}, \ldots, \overline{v_nv_1}$, gdje točke nazivamo vrhovima, a dužine bridovima. Dva brida koja imaju zajednički vrh nazivaju se *susjednim bridovima*. Poligon je jednostavan ako se sijeku samo susjedni bridovi. Poligon je jednostruko povezan ako se svaka jednostavna zatvorena krivulja u poligonu može kontinuirano smanjivati do točke bez napuštanja poligona, inače je poligon višestruko povezan (sadrži rupe). Primjerice, ako prostor sadrži objekte tada je poligon koji prikazuje taj prostor višestruko povezan, inače je jednostruko povezan.

Dvije točke p_i i p_j u poligonu su *vidljive* ako sve točke duž pravocrtne linije između p_i i p_j pripadaju tom poligonu. Ova binarna relacija naziva se *relacijom vidljivosti* i označavat ćemo je s $Vidljivo(p_i, p_j)$. Mijenjajući domenu relacije vidljivosti, može se definirati nekoliko važnih geometrijskih struktura. Primjerice, ograničavajući domenu na skup vrhova poligona, konstruiran je *graf vidljivosti*, u kojemu svaki čvor odgovara vrhu poligona i postoji brid između dvaju čvorova ako su odgovarajući vrhovi poligona vidljivi. Tretirajući p_i i p_j kao varijable, dobije se *poligon vidljivosti*. Poligon vidljivosti od p_i može se izraziti kao $VP_i = \{p_j | Vidljivo(p_i, p_j)\}$.

Prostorna ograničenja

Iz svakog položaja mjerenja q_i mjeri se udaljenost u n jednoliko raspodijeljenih smjerova. Mjerni podaci udaljenosti koji su dobiveni iz položaja mjerenja q_i zapisuju se kao $\mathcal{D}_i = \{d_{i,1}, d_{i,2}, \ldots, d_{i,n}\}$. Kut između dvaju uzastopnih smjerova očitanja je $\Delta\varphi = 2\pi/n$. Dodavanjem koordinatnog sustava (X_i,Y_i) na položaj mjerenja, gdje se os X_i podudara s prvim smjerom očitanja, j-ta komponenta od $\mathcal{D}_i$ odgovara točki uzorka $r_{i,j}$ s polarnim koordinatama $(d_{i,j}, j\Delta\varphi)$ koja se nalazi na bridu prostora. Transformacijom iz polarnih koordinata u Kartezijeve koordinate lako se otkriva niz bliskih, kolinearnih točaka uzoraka. Tada su mjerni podaci laserskog senzora udaljenosti skup od n točaka prostora, $\mathcal{R}_i = \{r_{i,k} \mid k = 1, \ldots, n\}$, gdje se točka $r_{i,k}$ može prikazati u Kartezijevom koordinatnom sustavu. Kako su mjerni podaci udaljenosti diskretna reprezentacija kontinuiranog poligona vidljivosti, mora se paziti na teoriju uzorkovanja. Zato se izvodi kriterij nalik Shannonovom teoremu uzorkovanja [27].

Neka je $S_m = (r_1, r_2, \ldots, r_m)$, $m \geq 3$, *serija* (podskup) od $\mathcal{R}$, gdje je $r_i = (x_i,y_i)$. Kaže se da serija u kojoj su sve točke uzoraka kolinearne ima *svojstvo kolinearnosti*. Ako dvije uzastopne točke uzoraka r_i i r_{i+1} imaju udaljenost Δr_i manju od unaprijed postavljene vrijednosti Δr, onda su one *blizu*, inače su *daleko*. Kaže se da serija u kojoj su svake dvije uzastopne točke uzoraka blizu ima *svojstvo blizine*. Spajanje svojstva blizine i kolinearnosti naziva se *BK svojstvom*. Zato se serije koje imaju *BK* svojstvo nazivaju *BK-serije*. Za danu vrijednost varijable Δr želi se okarakterizirati stvarni prostor u kojemu točke uzoraka pripadaju istoj *BK*-seriji ako leže na istom prostornom bridu. Kako je prikazano na slici 4.3, kod onih prostora koji nisu kompatibilni s vrijednošću varijable Δr mogu se pojaviti prividne *BK*-serije.



Slika 4.3. *Dvije prividne BK-serije.*

Neka je d_c oznaka za najmanju Euklidsku udaljenost između dviju točaka koje se nalaze na različitim najbližim nesusjednim bridovima u prostoru. Može se tvrditi da se nikad neće pojaviti prividne *BK*-serije, ako prostor zadovoljava uvjete u Teoremu 4.1.

Teorem 4.1. *Ako je* $d_c \geq 2\Delta$r *, tada sve točke uzoraka u nekoj BK-seriji leže na istom prostornom bridu.*

Dokaz: Neka e_p i e_r označavaju dva nesusjedna brida na udaljenosti d_c, a p i r točke na odgovarajućem bridu. Tada e_p i e_r mogu, ali i ne moraju imati zajednički susjedni brid. Pokazat će se da teorem vrijedi u oba slučaja.

i) *bridovi imaju zajednički susjedni brid;*

Neka e_q označava zajednički susjedni brid, a u i v kutove koje e_q definira sa e_p, odnosno e_r. Dužina brida e označena je s $|e|$, dok je točka sjecišta linijskog segmenta $\overline{pr}$ i brida

e_q označena sa q. Neka se sa d_{pq} i d_{qr} označavaju udaljenosti između odgovarajućih točaka. Situacija je prikazana na slici 4.4(a), gdje se pretpostavlja da je $u \leq v$. Sa slike 4.4(a) da se zaključiti da $\max(d_{pq}, d_{qr}) \geq d_c/2$, pri čemu jednakost vrijedi uz sljedeće uvjete:

1) $u = v$,

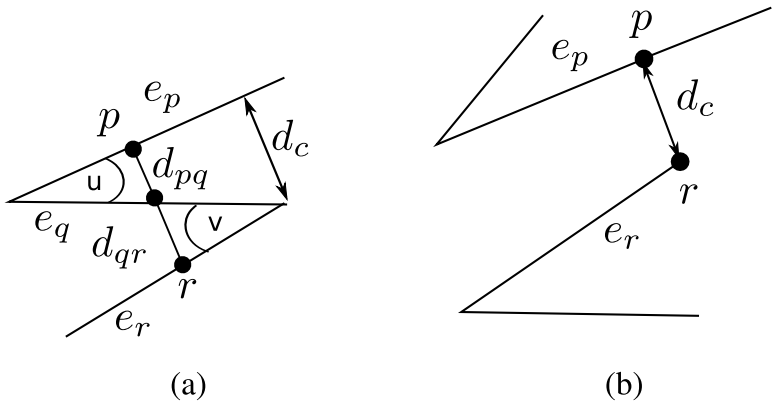
2) $|e_p| \geq |e_q| \cos(u)/2$ i

3) $|e_r| \geq |e_q| \cos(v)/2$.

Kako je $max(d_{pq}, d_{qr}) \geq d_c/2 \geq \Delta r > \Delta r_i$, niti jedna od tri točke (jedna sa svakog brida) ne može definirati BK-seriju.

ii) *bridovi nemaju zajednički susjedni brid;*

Ovaj je slučaj ilustriran na slici 4.4(b). Kako je $d_c > \Delta r$, kombinacija točaka sa različitih bridova ne može definirati BK-seriju.



Slika 4.4. *Dva slučaja najbližih nesusjednih bridova.*

Kako Teorem 4.1 nameće donju granicu na udaljenost između nesusjednih bridova poligona, n-stranični poligoni (gdje je $n \geq 4$) u prostoru ne mogu biti proizvoljno mali. No, kako su kod poligona s tri stranice (trokuti) svi bridovi susjedni, trostranični poligoni mogu biti proizvoljno mali a da ne krše uvjete Teorema 4.1. Nadalje, da bi se izbjegli proizvoljno mali objekti, svaki objekt u prostoru treba biti dovoljno velik da pokrije krug promjera Δr.

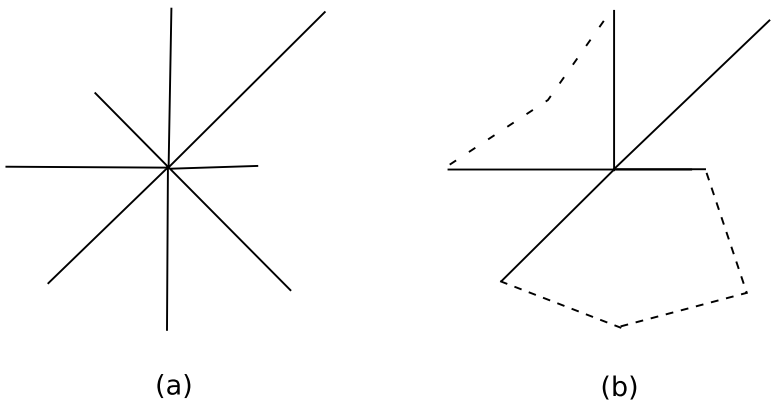
Poligoni mjerenja

Poligon mjerenja stvoren iz i-tog skupa mjernih podataka $\mathcal{R}_i$, označava se sa P_i. Granica se poligona mjerenja P_i predstavlja uređenim parom (L_i,B_i), gdje $L_i = \{l_{i,j}\}$ sadrži istražene dijelove prostornog brida, a $B_i = \{b_{i,k}\}$ sadrži granične bridove. Radi jednostavnosti, elementi od L_i i B_i nazivaju se l-bridovima odnosno b-bridovima. Dva su

poligona mjerenja P_i i P_j jednaka, ako vrijedi $L_i = L_j$, što se označava s $P_i = *P_j$. Granica vjernog poligona mjerenja P_F uređeni je par koji sadrži prazan skup B_F i zapisuje se kao $P_F = (L_F,\emptyset)$.

Poligon se mjerenja stvara iz skupa mjernih podataka udaljenosti u dva koraka. Prvo se skup mjernih podataka udaljenosti transformira u poligon zvjezdastog oblika, a zatim se određene operacije (ekspanzija i fuzija) izvode nad tim zvjezdastim poligonom da bi ga transformirali u poligon mjerenja.

Da bi se stvorio zvjezdasti poligon, svake se dvije uzastopne točke uzoraka r_i i r_{i+1} spajaju parom (b_i,b'_i) susjednog b-brida, čija zajednička točka $p_{i,i'}$ leži na infinitezimalnoj udaljenosti od točke s koje je izvršeno mjerenje (točka mjerenja) i gdje se nesusjedni b-bridovi ne sijeku (slika 4.5.(a)). Zvjezdasti poligon ima infinitezimalnu površinu, centriran je oko točke mjerenja i ima po jedan vrh za svaku od n točaka uzorka i n vrhova na infinitezimalnoj udaljenosti od točke mjerenja.



Slika 4.5. *(a) Inicijalni i (b) prošireni zvjezdasti poligon.*

Da bi se zvjezdasti poligon proširio (ekspanzija), prvo se definira krug infinitezimalnog polumjera za svaki par b-bridova $(b_i,b_{i'})$ koji se nalazi na konkavnoj strani od $(b_i,p_{i,i'},b_{i'})$ i s jednom zajedničkom točkom s b_i, odnosno $b_{i'}$. Zatim se krug počinje širiti (tj. povećava se polumjer kruga) tako da i dalje ima jednu zajedničku točku s b_i odnosno $b_{i'}$. Ekspanzija završava kad:

i) daljnje širenje kruga prouzroči da krug siječe neki brid u dvije točke;

ii) krug siječe a) dvije točke uzoraka r_j i $r_{j'}$; b) dva brida l_k i $l_{k'}$ ili c) točku uzorka r_j i brid l_k;

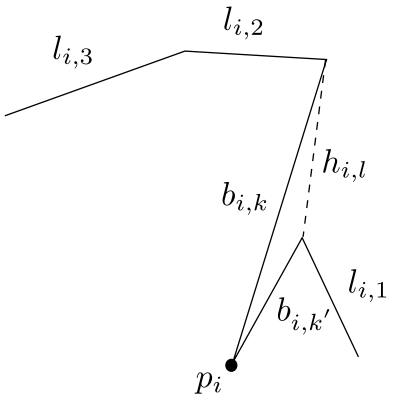
iii) promjer kruga je postao jednak Δr.

Za svaki uvjet završetka ekspanzije izvode se odgovarajuće operacije. U slučaju i) briše se svaki b-brid koji pripada granici slobodnog prostora i koji siječe krug. Zatim se ekspanzija kruga nastavlja. U slučaju ii.a) ako neki drugi krug siječe (1) točku uzorka

r_m i (2) bilo koju od r_j i $r_{j'}$; i r_j, $r_{j'}$ i r_m kolinearne, brišu se oba kruga te se svaki par b-bridova koji spaja točke uzoraka zamjenjuje l-bridom, spajajući točke uzoraka. Ako nijedan drugi krug ne siječe ni točku uzorka r_m ni bilo koju od r_j ili $r_{j'}$, krug se briše, a svi ostali krugovi dostignu maksimalno proširenje. U slučaju ii.b) krug se briše, b-bridovi b_i i $b_{i'}$ se preimenuju s "l", dok se točka $p_{i,i'}$ prenosi u točku sjecišta bridova l_k i $l_{k'}$. U slučaju ii.c), krug se briše, te ako je točka uzorka r_j kolinearna s bridom l_k, bridovi b_i i $b_{i'}$ zamjenjuju se bridom l_k, spajajući nezajedničke krajnje točke od b_i i $b_{i'}$. Primjer proširenog zvjezdastog poligona prikazan je na slici 4.5(b).

Nakon što su svi krugovi maksimalno prošireni (te sustavno obrisani) slijedi fuzija. Kolinearni bridovi l_k i $l_{k'}$ sa zajedničkom krajnjom točkom zamjenjuju se bridom l_k definiranim između nezajedničkih krajnjih točaka od l_k i $l_{k'}$. Izvodeći maksimalan broj fuzija, dobiva se poligon mjerenja.

Poligon se mjerenja transformira u *prošireni poligon mjerenja* EP_i, ako se za svaki par b-bridova (b_i,$b_{i'}$) može definirati brid h_i (h_i je aproksimacija skočnog brida) između nezajedničkih krajnjih točaka od b_i i $b_{i'}$. Granica se proširenog poligona mjerenja predstavlja uređenom trojkom (L_i,B_i,H_i), gdje je L_i=$\{l_{i,j}\}$, B_i=$\{b_{i,k}\}$ i H_i=$\{h_{i,l}\}$.



Slika 4.6. *Dio proširenog poligona mjerenja.*

Na slici 4.6 pretpostavljeno je da linijski segment $l_{i,2}$ ima jednu bližu i jednu udaljenu krajnju točku (pretpostavlja se da su linijski segmenti prikazani na slici stvoreni na osnovi skupa mjernih podataka udaljenosti koji je dobiven iz pozicije p_i).

Poligoni istraživanja

Ovdje se uvodi koncept *poligona istraživanja* $P_E(i)$ i *proširenog poligona istraživanja* $EP_E(i)$. Granica poligona istraživanja $P_E(i)$ predstavlja se uređenim parom $(L_E(i), B_E(i))$ i stvara se dodavanjem bridova poligona mjerenja P_i poligonu istraživanja $P_E(i-1)$, gdje je $P_E(0)$ prazan poligon i označava se sa $\emptyset$. Ako b-brid od $P_E(i-1)$ siječe b-brid od P_i, tada se definiraju četiri para b-bridova, gdje svih osam b-bridova ima zajedničku točku. Na kraju slijede ekspanzija i fuzija kako je prethodno opisano.

Granica proširenog poligona istraživanja $EP_E(i)$ predstavlja se uređenom trojkom $(EL_E(i), EB_E(i), EH_E(i))$ i stvara se uvođenjem h-bridova između nezajedničkih krajnjih točaka svakog para b-bridova iz $P_E(i)$.

Graf istraživanja

Da bi se olakšalo planiranje putanje robota koristi se *graf istraživanja* $\mathcal{G}_E(i)$. $\mathcal{G}_E(i) = (\mathcal{N}_E, \mathcal{E}_E)$ je spojeni, neusmjereni graf, gdje su $\mathcal{N}_E = \{n_i\}$ skup *čvorova* i $\mathcal{E}_E = \{e_{i,j}\}$ skup *bridova*. Čvorovi odgovaraju položajima mjerenja dok su bridovi definirani između vidljivih čvorova. Nadalje, čvorovi n_i imaju pridružene vrijednosti dobitka g_i, a bridovi e_{ij} imaju pridružene vrijednosti cijena c_{ij}. Vrijednost dobitka predstavlja procijenjen dobitak informacija na osnovi mjerenja u čvoru, dok vrijednost cijene predstavlja cijenu gibanja robota između pozicija koje odgovaraju krajnjim točkama bridova.

Graf istraživanja $\mathcal{G}_E(i)$ dobiva se iz grafa istraživanja $\mathcal{G}_E(i-1)$, gdje je $\mathcal{G}_E(0) = \{(0,0), \emptyset\}$ te iz proširenog poligona istraživanja $EP_E(i)$ kako je opisano u nastavku. Prvo se grafu $\mathcal{G}_E(i)$ dodjeljuje graf $\mathcal{G}_E(i-1)$. Zatim se za svaki h-brid u $EP_E(i)$ u $\mathcal{G}_E(i)$ dodaje čvor koji se definira s odgovarajućim kandidatom za sljedeću poziciju mjerenja. Nakon toga se provjeravaju relacije vidljivosti za čvorove u $\mathcal{G}_E(i)$ te se u skladu s tim dodaju novi bridovi. Na kraju se ažuriraju vrijednosti dobitka i cijene u $\mathcal{G}_E(i)$.

Planiranje putanje

Koristi se jednostavna strategija odabira čvora koji se sljedeći mora posjetiti u grafu istraživanja $\mathcal{G}_E(i)$. Svaki je čvor u $\mathcal{G}_E(i)$ posjećen ili neposjećen. Čvor je posjećen ako je robot obavio mjerenje s tog čvora. Inače je čvor neposjećen. Neposjećeni čvorovi predstavljaju kandidate za točke mjerenja. Za odabir točaka mjerenja između neposjećenih čvorova koristi se kriterijska funkcija $C(n_i, n_j)$. Kako je cilj izvesti strategiju istraživanja koja pod određenim uvjetima jamči potpuno istraživanje prostora unutar konačnog broja mjerenja, kriterijska funkcija zadovoljava taj cilj, ali bez tvrdnje da je optimalan u smislu najmanjeg broja mjerenja koja su potrebna da se istraži prostor.

Kako se nova mjerenja robota odvijaju ispred prethodno neistraženih područja, dobit će se nove informacije. U skladu s tim, poligon istraživanja će nakon ažuriranja konvergirati prema vjernom prikazu poligona. Posebice, opseg poligona istraživanja konvergira prema opsegu vjerno prikazanog poligona. Na konvergiranju opsega poligona zasnivaju se vrijednosti dobitka čvora te se najveća vrijednost dobitka pridaje onom čvoru čiji poligon mjerenja daje najveće proširenje opsega poligona istraživanja. Kako je stvarno povećanje opsega *a priori* nepoznato, za procjenu iznosa novih informacija dobivenih iz mjerenja koristi se heuristika. Korištenje heuristike u vrijednosti dobitka čvora opravdava se činjenicom da je potreban pravilan redoslijed kandidata za novu točku mjerenja.

Neka $S = h_{i,k}$ označava skup h-bridova koji su vidljivi iz točke mjerenja p_{i-1}. Svaki vidljivi h-brid $h_{i,k}$ definira trokut s odgovarajućim kandidatom za novu točku mjerenja

$p_{i,k}$. Unutarnji se kut $\alpha_{i,k}$ trokuta u vrhu $p_{i,k}$ definira kao $\alpha_{i,k} = \sphericalangle\,(p_{i,k}, h_{i,k})$. Kao primjer, na slici 4.7 poligon mjerenja P_{i-1} sadrži dva h-brida i sukladno su definirana dva kandidata za novu točku mjerenja $p_{i,1}$ i $p_{i,2}$. Vrijednost dobitka g_{i-1} čvora n_{i-1} definirana je kao zbroj kutova koji su definirani svim h-bridovima vidljivima iz p_{i-1}

$$g_i = \sum_k \alpha_{i,k}. \tag{4.1}$$

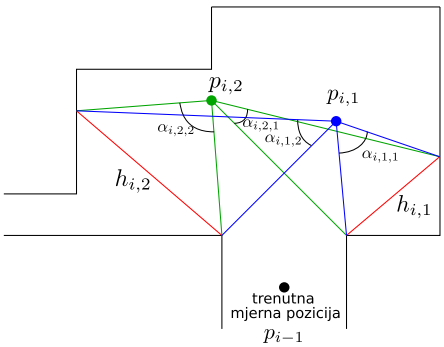
Vrijednost cijene c_{ij} brida e_{ij} definira se kao Euklidska udaljenost između pozicija koje odgovaraju krajnjim točkama brida:

$$c_{ij} = \|n_i - n_j\|. \tag{4.2}$$

Kao kriterij za odabir čvora koristi se *odnos dobitka i cijene* u trenutnom čvoru mjerenja n_i:

$$C(n_i, n_j) = \frac{g_j}{c_{i,j}}. \tag{4.3}$$

Ako čvor n_j nije vidljiv iz čvora n_i, cijena se računa kao zbroj cijena vidljivih bridova duž "najjeftinije" putanje između n_i i n_j. Čvor s najvećim odnosom dobitka i cijene izabire se kao nova točka mjerenja.



Slika 4.7. *Računanje vrijednosti dobitka čvora.*

Algoritam istraživanja prostora

Pseudokod algoritma istraživanja dan je algoritmom 4.1. Inicijalno, čvorovi n_0 i n_1 predstavljaju istu poziciju, tako da procedura **idi-na-novu-mjernu-poziciju** ne radi ništa tijekom prve iteracije, dok se poligonu istraživanja dodjeljuje prazan poligon, a graf istraživanja sadrži čvor koji odgovara početnoj poziciji. Algoritam pretraživanja

prostora prestaje nakon k-tog mjerenja ako svi čvorovi imaju iznos dobitka jednak nuli.

Algoritam 4.1: Ekmanova strategija istraživanja

$n_0 \leftarrow (0,0)\{Inicijalizacija\}$
$n_1 \leftarrow (0,0)$
$P_E(0) \leftarrow \varepsilon$
$\mathcal{G}_E(0) \leftarrow ((0,0),\emptyset)$
$i \leftarrow 1$
ponavljaj
  idi-na-novu-mjernu-poziciju(n_i)
  $\mathcal{R}_i \leftarrow$ **novo-mjerenje**
  $P_i \leftarrow$ **stvori-poligon-mjerenja**$(\mathcal{R}_i)$
  $P_E(i) \leftarrow$ **ažuriraj-poligon-istraživanja**$(P_E(i-1),\, P_i)$
  $EP_E(i) \leftarrow$ **stvori-prošireni-poligon-istraživanja**$(P_E(i))$
  $\mathcal{G}_E(i) \leftarrow$ **stvori-graf-istraživanja**$(\mathcal{G}_E(i-1),\, EP_E(i))$
  $n_{i+1} \leftarrow$ **odaberi-čvor**$(n_i,\, \mathcal{G}_E(i))$
  $i \leftarrow i+1$
dok nije $g_i = 0$

Lema 4.1 izražava kriterij za pronalazak trenutka kada prošireni poligon istraživanja postaje vjeran karti prostora.

Lema 4.1. *Ako neprazan prošireni poligon istraživanja* EP_E(i) *ne sadrži skočne bridove tada je on jednak vjernom prikazu poligona* P_F. *To jest,* $EH_E(i) = \emptyset \Rightarrow EP_E(i) = *P_F$.

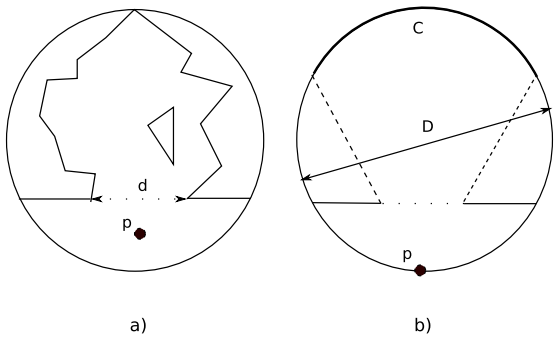
Dokaz: Pretpostavimo $EP_E(i) \neq *P_F$, ali $EH_E(i) = \emptyset$. Ako vrijedi $EP_E(i) \neq P_F$, tada postoji dio brida l koji pripada bridu u L_F, ali ne bilo kojem bridu u $EL_E(i)$. Bilo koji segment brida koji nije u $EL_E(i)$ je neistražen. Kako je svaki neistraženi dio brida odvojen skočnim bridom od slobodnog prostora, l mora isto biti odvojen skočnim bridom od slobodnog prostora. Slijedi da je $EH_E(i) \neq \emptyset$ te imamo kontradikciju.

Opaža se da implikacija u suprotnom smjeru nije moguća. To jest $EP_E(i) = P_F$ ne implicira nužno $EH_E(i) = \emptyset$. Iz leme 4.1 te gornje diskusije zaključuje se da se izvodi *previše* mjerenja, ali nikad *premalo*. Sljedeći korolar, koji odmah slijedi lemu 4.1, izražava kriterij brisanja iz grafa.

Korolar 4.2. *Ako najobećavajući čvor* n_{i+1} *u* $\mathcal{G}_E(i)$ *ima nulu za vrijednost dobitka* g_{i+1} *tada je prošireni poligon istraživanja vjeran karti prostora.*

Nakon uvođenja kriterija za detekciju trenutka kada je poligon istraživanja postao vjerna karta prostora ostaje za pokazati da će za bilo koji konačan prostor kod kojega vrijedi ranije opisano prostorno ograničenje biti stvorena vjerna karta iz konačnog broja mjerenja. Uvjet konačnosti broja mjerenja implicira postojanje najmanjeg kruga C polumjera D takvog da je $P_F \subset C$. Svako mjerenje osim prvog izvodi se s udaljenosti d_m od h-brida. Nadalje, h-bridovi se mogu promatrati kao prozori između istraženog i neistraženog područja. Širina d bilo kojeg prozora h-brida ima vrijednost u intervalu $[\Delta r, D]$. Primjer mjerenja ispred prozora h-brida veličine d prikazan je na slici 4.8(a).

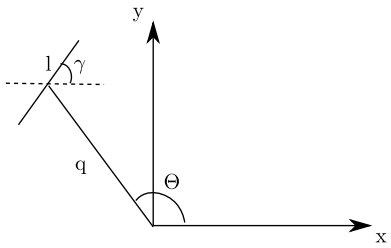
Lema 4.2. *Uz dani* $(\Delta s,\, D)$ *i prostor za koji vrijede prethodno opisana ograničenja*



Slika 4.8. *(a) prozor h-brida širine d, i (b) maksimalno područje C koje se može vidjeti kroz prozor h-brida.*

prostora, postoji neki neinfinitezimalan $\Delta\varphi$ *takav da je najmanje jedan brid prostora koji je iza prozora h-brida prekriven BK-serijom.*

Dokaz: Neka je proizvoljan brid prostora, prikazan kao uređena četvorka (q, l, γ, θ), izražen u Kartezijevom koordinatnom sustavu i centriran oko točke mjerenja, koji ima X-os paralenu prozoru h-brida, kako je prikazano na slici 4.9. Najveći kutni dodatak $\Delta\varphi$, koji ima mogućnost prekrivanja najmanje jednog brida BK-serijom, smanjuje se u sljedećim slučajevima: 1) q se povećava, 2) l se smanjuje, ili 3) $(\theta - \gamma)$ se smanjuje. Pokazat će se da je iznos povećanja/smanjenja svakog parametra ograničena, te da je tada najveći $\Delta\varphi$ koji se može koristiti za pokrivanje najmanje jednog brida BK-serijom neinfinitezimalan.



Slika 4.9. *Proizvoljan brid prostora prikazan kao uređena četvorka* (q, l, γ, θ).

Pretpostavlja se da se robot nalazi na poziciji p ispred prozora h-brida, na granici kruga promjera D (slika 4.8(b)). Kako je D konačan, q ima gornju granicu. Područje ograničeno s dvije crtkane linije, kružnim segmentom C, i točkastom linijom, odgovara najvećem području koje robotu može biti vidljivo kroz prozor h-brida. Pod pretpostavkom da je prostor zatvoren, linija između p i bilo koje druge pozicije na C siječe barem jedan brid. Ako se dijelovi bridova koji nisu vidljivi robotu ignoriraju, linija između p i bilo koje druge pozicije na C sjeći će točno jedan brid. Iz Teorema 4.1 slijedi da bilo koja dva dijela brida koja ne pripadaju susjednim bridovima moraju imati duljinu koja je

veća od $2\Delta r$. U skladu s tim područje sadrži konačan broj bridnih dijelova. Da bi linija između p i bilo koje druge točke na C sjekla vidljiv dio brida, niti jedan l niti $(\theta - \gamma)$ ne može biti proizvoljno malen za sve dijelove brida. Kako je povećanje/smanjenje svih parametara ograničeno, najveći $\Delta\varphi$ koji se može koristiti za pokrivanje najmanje jednog brida s BK-serijom, je neinfinitezimalan.

Lema 4.3. *Euklidska udaljenost između krajnjih točaka bilo koje BK-serije je najmanje* $2d_m \sin(\Delta\varphi)$.

Dokaz: Odabirom kandidata za novu točku mjerenja udaljenost između svakog prethodno neistraženog dijela brida i bilo kojeg kandidata za novu točku mjerenja prelazi vrijednost d_m. Zato je udaljenost između krajnjih točaka BK-serije najmanje $2d_m \sin(\Delta\varphi)$, odgovarajući BK-seriji duljine 3. Druga točka uzorka odgovara smjeru očitanja koji je ortogonalan na brid prekriven BK-serijom.

Teorem 4.3. *Za dane* $(\Delta s,\ D)$ *i prostor s opsegom* k, *koji zadovoljava prethodno navedena ograničenja prostora, postoji neki neinfinitezimalni kutni dodatak $\Delta\varphi$, takav da je broj mjerenja potrebnih da bi se potpuno istražio prostor, tj. da algoritam konvergira, konačan.*

Dokaz: Postojanje takvog $\Delta\varphi$ jamči lema 4.2. Iz leme 4.3 slijedi da nikakav dodatak od $L_E(i)$ ne može biti manji od $2d_m \sin(\Delta\varphi)$. Pretpostavljajući opseg k nije potrebno više mjerenja od $k/2d_m \sin(\Delta\varphi)$. Kako je ta vrijednost konačna, algoritam očito konvergira.

4.2 Izdvajanje linijskih segmenata iz senzorskih mjerenja

Izdvajanje značajki i grupiranje podataka važno je za primjene vezane za lokalizaciju mobilnih robota i izgradnju karte prostora. Pronalaženje linija koje dobro opisuju mjerne točke senzora udaljenosti znatno smanjuje složenost reprezentacije podataka. U literaturi postoje brojne metode koje izdvajaju linije iz senzorskih podataka [11], [119], [108]. Neki autori koriste Houghovu transformaciju za pronalaženje linija koje opisuju laserske ili sonarske podatke, ali sama Houghova transformacija ne uzima u obzir nesigurnost i šum pri procjenjivanju parametara linija. Pristup zasnivan na Kalmanovom filtru [106] dopušta samo uniformno otežavanje doprinosa točaka u opisu linija. Pfister *et al.* [94] razmatraju točno pridjeljivanje linija skupu točaka koje posjeduju određenu nesigurnost. Njihova metoda zasniva se na otežanom utjecaju svake točke na cjelokupni postupak pridjeljivanja linija s obzirom na nesigurnost točaka, koja je dobivena iz modela zašumljenog senzora.

U ovom je radu primijenjen algoritam izdvajanja linijskih segmenata prema [94], koji objedinjuje utjecaj nesigurnosti mjerenja i pozicije mobilnog robota. Algoritam se zasniva na Houghovoj transformaciji, otežanoj metodi najmanjih kvadrata i χ^2-testu (hi-kvadrat).

Houghova transformacija (HT) dobro je poznata metoda izlučivanja značajki iz slike

[55]. Razmatrajući samo linijske segmente kao značajke, HT pronalazi linije koje prolaze kroz skup kolinearnih rubnih točaka u slici. Veliki broj računskih operacija glavni je nedostatak primjene te metode.

Metode korištene u problemu istraživanja nepoznatog prostora, moraju biti pogodne za rad u stvarnom vremenu, dok robot istražuje prostor. Zato se u radu koristi vjerojatnosna Houghova transformacija (VHT) s naknadnom obradom [79]. VHT koristi samo dio slučajno odabranih rubnih točaka u slici čime značajno skraćuje vrijeme izračuna potrebno za detekciju linija. Također, VHT odmah nalazi linijske segmente za razliku od standardne HT koja nalazi linije.

Grupiranje, odnosno spajanje, značajki sličnih atributa potrebno je da bi se detektirale linije koje odgovaraju istom prostornom zidu. Različite metode spajanja djelomično preklapajućih linijskih segmenata izložene su u [128]. U ovome se radu za spajanje sličnih linija koristi metoda zasnovana na χ^2-testu. Precizno modeliranje nesigurnosti podataka ključno je za problem grupiranja odnosno spajanja značajki, budući da je test hipoteza zasnivan na χ^2 metodi ispravan onoliko koliko je ispravno modelirana nesigurnost.

Opis pojmova iz teorije vjerojatnosti i statistike koji se koriste u nastavku, može se naći u [6].

Postupak pronalaženja linijskih segmenata

U nastavku je opisano kako se iz laserskih očitanja dobivaju linijski segmenti. Mjerni su podaci laserskog senzora udaljenosti skup od n točaka, $\mathcal{R} = \{r_k \mid k = 1, \ldots, n\}$, gdje je r_k k-to očitanje u Kartezijevim koordinatama. Cilj je ovog postupka detektirati i grupirati točke iz skupa očitanja $\mathcal{R}$ u M podskupova tako da svaki podskup sadrži točke koje su kolinearne unutar nekakve granice pogreške. Pri tome M nije predodređen broj. Za svaki se podskup određuje optimalni linijski segment S s pripadnom matricom kovarijanci Σ_S. Određivanje je linijskih segmenata iz laserskih očitanja iterativan postupak koji se sastoji od sljedećih koraka:

(1) Početna procjena linijskih segmenata. Iz skupa točaka $\mathcal{R}$ određuju se početne procjene linijskih segmenata pomoću vjerojatnosne Houghove transformacije.

(2) Grupiranje točaka. Za svaki linijski segment $\hat{S}_i$ dobiven u prethodnom koraku određuje se podskup točaka $\mathcal{R}^i_{\hat{S}}$ iz skupa $\mathcal{R}$.

(3) Modeliranje šuma točke senzorskog mjerenja. Za svaki skup $\mathcal{R}^i_{\hat{S}}$ računaju se matrice kovarijanci točaka tog skupa Σ_{r_k}.

(4) Otežana metoda pronalaženja linijskog segmenta. Za svaki skup $\mathcal{R}^i_{\hat{S}}$ računa se optimalni linijski segment $S_i = [\alpha, \rho, \psi_a, \psi_b]$ pomoću pripadnih matrica kovarijanci točaka Σ_{r_k}.

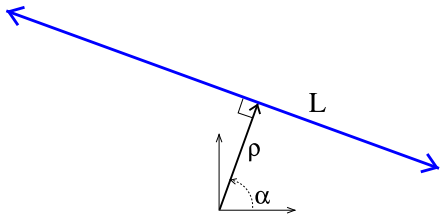
(5) Procjena kovarijanci linija. Za svaki linijski segment S_i procjenjuje se matrica kovarijanci linije Σ_L^i koja odgovara tom segmentu.

(6) Spajanje sličnih linijskih segmenata. Svi parovi dobivenih linijskih segmenata u ovom i predhodnim mjerenjima pokušavaju se spojiti u jedan linijski segment korištenjem χ^2-testa.

4.2.1 Početna procjena linijskih segmenata

Reprezentacija linija

Polarna reprezentacija linije omogućuje lako uspoređivanje orijentacije i pozicije linije. Linija u polarnom zapisu definirana je kao:



Slika 4.10. *Polarna reprezentacija linije.*

$$L = \begin{bmatrix} \rho \\ \alpha \end{bmatrix}, \tag{4.4}$$

gdje su ρ i α duljina i kut vektora normale na liniju. Na slici 4.10 prikazana je linija u polarnom zapisu. Jednadžba linije dana je sljedećim izrazom:

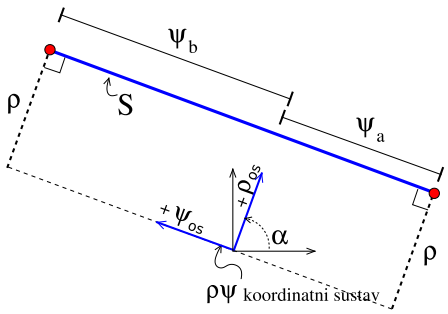
$$x\cos(\alpha) + y\sin(\alpha) = \rho. \tag{4.5}$$

Reprezentacija linijskih segmenata

Reprezentacija linijskog segmenta zasniva se na reprezentaciji linije i pridruživanju krajnjih točaka liniji. Krajnje točke linijskog segmenta predstavljene su skalarnim vrijednostima ψ_a i ψ_b (slika 4.11). Za liniju orijentacije α definiran je ρ-ψ koordinatni sustav dobiven rotiranjem x-y koordinatnog sustava za kut α. Skalarne vrijednosti ψ_a i ψ_b mjere se u odnosu na ψ os i mogu poprimiti i pozitivne i negativne vrijednosti.

Linijski segment definiran je s:

$$S = \begin{bmatrix} \alpha \\ \rho \\ \psi_a \\ \psi_b \end{bmatrix}. \tag{4.6}$$

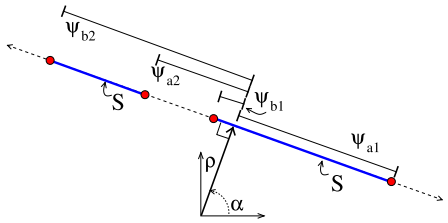


Slika 4.11. *Reprezentacija linijskog segmenta*

Linijski segment ima mogućnost višestrukog zapisa svih linijskih segmenata koji se nalaze na istoj liniji, kao što je prikazano na slici 4.12:

$$S = \begin{bmatrix} \alpha \\ \rho \\ \psi_{a1} \\ \psi_{b1} \\ . \\ . \\ . \\ \psi_{an} \\ \psi_{bn} \end{bmatrix}, \tag{4.7}$$

gdje je n broj linijskih segmenata koji se nalaze na istoj liniji.



Slika 4.12. *Dva linijska segmenta na istoj liniji.*

Houghova transformacija

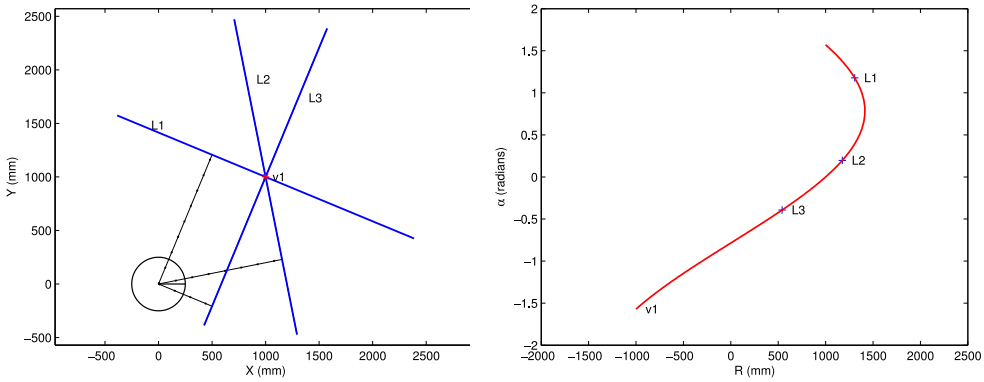
Houghova transformacija (HT) je često korišten algoritam za detekciju značajki u računalnom vidu. Svrha tog algoritma je pronalazak značajki prema nekoj klasi definiranih značajki primjenom procedure glasovanja. Procedura glasovanja odvija se u Houghovom prostoru, gdje se kandidati određuju kao lokalni maksimumi. U primjeni HT za otkrivanje linija svaka je točka $\{r_k\}$, $k = 1, \ldots, n$ transformirana u diskretiziranu krivulju u Houghovom prostoru i pohranjena u polje, tzv. akumulator.

Razmotrimo liniju L u polarnom zapisu. Parametri ρ i α definiraju dimenzije Houghova prostora. Svaka točka iz skupa $\mathcal{R}$ prikazana je Kartezijevim koordinatama $r_k = [x_k, y_k]$. Houghov je prostor definiran kao dvodimenzionalno polje s indeksima (koordinatama) i i j. Svaka koordinata definira diskretne vrijednosti $\alpha(j)$ i $\rho(i)$. $\alpha(j)$ je diskretizirana po koracima $\Delta\alpha$ u intervalu $[-\frac{\pi}{2}, \frac{\pi}{2}]$, dok je $\rho(i)$ diskretiziran po koracima $\Delta\rho$ u intervalu $[0, S_{max}]$, gdje je S_{max} najveći domet senzora. Dakle $\{i, j\}$ koordinata u diskretiziranom Houghovu prostoru prikazuje linije s parametrima unutar ograda $\rho(i) \pm \Delta\rho/2$ i $\alpha(j) \pm \Delta\alpha/2$. Inicijalno je Houghov prostor prazan, odnosno popunjen nulama. Za svaku zadanu točku r_k moguće je izračunati liniju koja bi prolazila kroz tu točku i imala orijentaciju vektora normale $\alpha(i)$:

$$\rho_{ik} = x_k \cos(\alpha(i)) + y_k \sin(\alpha(i)) \tag{4.8}$$

Iz vrijednosti ρ_{ik} moguće je odrediti takav indeks j za koji je $\rho(j) - \frac{\Delta\rho}{2} < \rho_{ik} \leq \rho(j) + \frac{\Delta\rho}{2}$. Na svim se takvim koordinatama $\{i, j\}$ povećava vrijednost u Houghovu prostoru. Na kraju cijelog postupka koordinata u Houghovu prostoru s najvećom vrijednosti prikazuje liniju za koju je glasovalo najviše točaka.

Na slici 4.13, lijevo, prikazana je točka i tri potencijalne linije koje prolaze kroz točku, dok je na slici 4.13, desno, prikazan Houghov prostor s tri pripadne točke koje odgovaraju trima linijama. Skup svih linija koje prolaze kroz točku predstavljaju sinu-

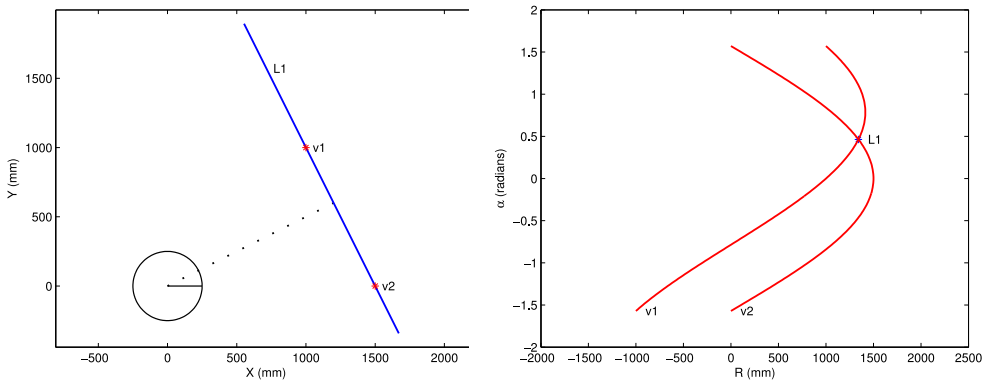


Slika 4.13. *Potencijalne linije kroz jednu točku u Kartezijevim koordinatama i u Houghovu prostoru.*

soidu u Houghovu prostoru. Na slici 4.14, lijevo, prikazane su dvije kolinearne točke i dobivena linija koja prolazi kroz te dvije točke, dok je na slici 4.14, desno, prikazana krivulja u Houghovu prostoru za obje točke te njihovo sjecište koje predstavlja dobivenu liniju.

Vjerojatnosna Houghova transformacija

Veliki broj računskih operacija glavni je nedostatak primjene standardne HT koji se znatno ublažava primjenom VHT koja koristi samo dijelić slučajno odabranih točaka.



Slika 4.14. *Dobivena linija kroz dvije točke u Kartezijevim koordinatama i Houghovom prostoru.*

VHT odmah nalazi linijske segmente za razliku od standardne HT koja nalazi linije.

Algoritam učestalo odabire novu točku za glasovanje. Nakon što se otkrije linija točke koje su glasovale za tu liniju više se ne razmatraju i tako algoritam nastavlja s manjim skupom točaka. Algoritam ne zahtijeva kriterij za zaustavljanje već završava kada su sve točke pridružene linijama. VHT ima manje lažno pozitivnih otkrivenih linija u odnosu na standardnu HT.

4.2.2 Grupiranje točaka

Neka je početna procjena linijskog segmenta označena sa $\hat{S}$, s parametrima $\hat{\rho}$ i $\hat{\alpha}$, a skup točaka koji odgovara tom segmentu označen s $\mathcal{R}_{\hat{S}}$. Skup $\mathcal{R}_{\hat{S}}$ određuje se tako što se računa odstupanje točaka od segmenta

$$\delta\rho_k = |x_k \cos\hat{\alpha} + y_k \sin\hat{\alpha} - \hat{\rho}|, \tag{4.9}$$

gdje je $r_k = [x_k, y_k]$ k-ta točka iz skupa $\mathcal{R}$. Točke koje imaju odstupanje $\delta\rho_k$ manje od granice $\Delta\rho$ dodaju se u skup $\mathcal{R}_{\hat{S}}$, gdje je $\Delta\rho$ korak diskretizacije koordinate ρ Houghova prostora.

4.2.3 Modeliranje šuma točke senzorskog mjerenja

Laserski senzor udaljenosti mjeri udaljenost do najbliže prepreke u $n = 361$ smjerova u rasponu od $\frac{-\pi}{2}$ do $\frac{\pi}{2}$. Skup $\mathcal{R}$ od n točaka iz jedne mjerne pozicije zapisan je u lokalnom koordinatnom sustavu robota. Zapišimo koordinate stvarnih prepreka u polarnom zapisu:

$$r_k = d_k \begin{bmatrix} \cos(\Theta_k) \\ \sin(\Theta_k) \end{bmatrix}, \tag{4.10}$$

gdje je d_k udaljenost od prepreke, a Θ_k kut u odnosu na pozitivnu poluos x. Mjerna vrijednost $\hat{d}_k$ sastoji se od stvarne vrijednosti udaljenosti d_k i dodatne pogreške zbog

šuma ϵ_d:

$$\hat{d}_k = d_k + \epsilon_{d_k}. \tag{4.11}$$

Analogno, mjerni kut $\hat{\Theta}_k$ sadrži stvarni kut Θ_k i pogrešku ϵ_Θ:

$$\hat{\Theta}_k = \Theta_k + \epsilon_{\Theta_k}. \tag{4.12}$$

Stoga, mjernu točku možemo prikazati kao:

$$\hat{r}_k = (d_k + \epsilon_{d_k}) \begin{bmatrix} \cos(\Theta_k + \epsilon_{\Theta_k}) \\ \sin(\Theta_k + \epsilon_{\Theta_k}) \end{bmatrix}. \tag{4.13}$$

Pretpostavimo da sve pogreške imaju Gaussovu razdiobu sa srednjom vrijednošću jednakom nuli i pripadnim varijancama σ_d^2 i σ_Θ^2. Uz pretpostavku da je $\epsilon_\Theta << 1°$ možemo uvesti sljedeću aproksimaciju:

$$\sin(\epsilon_\Theta) \approx \epsilon_\Theta, \tag{4.14}$$

$$\cos(\epsilon_\Theta) \approx 1. \tag{4.15}$$

Mjernu točku također možemo prikazati kao zbroj stvarne komponente i pogreške:

$$\hat{r}_k = r_k + \delta r_k. \tag{4.16}$$

Iz jednadžbi (4.13) i (4.16) možemo izraziti δr_k kao:

$$\delta r_k = (d_k + \epsilon_d)\epsilon_\Theta \begin{bmatrix} -\sin(\Theta_k) \\ \cos(\Theta_k) \end{bmatrix} + \epsilon_d \begin{bmatrix} \cos(\Theta_k) \\ \sin(\Theta_k) \end{bmatrix}. \tag{4.17}$$

Uz pretpostavku nezavisnosti ϵ_d i ϵ_Θ matrica kovarijanci mjerne točke je:

$$\Sigma_{r_k} = \frac{d_k^2 \sigma_\Theta^2}{2} \begin{bmatrix} 2\sin^2(\Theta_k)) & -\sin(2\Theta_k) \\ -\sin(2\Theta_k) & 2\cos^2(\Theta_k) \end{bmatrix} + \frac{\sigma_d^2}{2} \begin{bmatrix} 2\cos^2(\Theta_k) & \sin(2\Theta_k) \\ \sin(2\Theta_k) & 2\sin^2(\Theta_k) \end{bmatrix}. \tag{4.18}$$

Za praktične izračune $\hat{\Theta}_k$ i $\hat{d}_k$ su upotrebljeni umjesto Θ_k i d_k za procjenu matrice kovarijanci točke.

4.2.4 Otežana metoda pronalaženja linijskog segmenta

Ovdje su dani gotovi izrazi za računanje procjene parametara linija prema [94]. Izvodi navedenih izraza nalaze se u [93] i [92]. Nakon početne procjene vjerojatnosnom Houghovom transformacijom, računaju se parametri linija pomoću formulacije maksimalne vjerodostojnosti. Pritom se minimizira udaljenost točaka mjerenja od linija

$$\delta\rho_k = \hat{d}_k \cos(\hat{\alpha} - \hat{\Theta}_k) - \hat{\rho}. \tag{4.19}$$

Procjena parametra ρ je

$$\hat{\rho} = \frac{\sum_{k=1}^{n} \frac{\hat{d}_k \cos(\hat{\alpha} - \hat{\Theta}_k)}{\Sigma_{\delta\rho_k}}}{\sum_{k=1}^{n} \frac{1}{\Sigma_{\delta\rho_k}}}, \tag{4.20}$$

gdje je $\Sigma_{\delta\rho_k}$

$$\Sigma_{\delta\rho_k} = \sigma_d^2 cos^2(\hat{\alpha} - \hat{\Theta}_k) + \sigma_\Theta^2 d_k^2 \sin^2(\hat{\alpha} - \hat{\Theta}_k), \tag{4.21}$$

varijanca udaljenosti $\delta\rho_k$ koja se minimizira. Neka je $\hat{\psi}_k$ projicirana točka mjerenja na liniju gledano u $\rho - \psi$ koordinatnom sustavu:

$$\hat{\psi}_k = \hat{d}_k \sin(\hat{\alpha} - \hat{\Theta}_k). \tag{4.22}$$

Centar rotacijske nesigurnosti objašnjen je u odlomku 4.2.5, a ovdje navodimo njegovu procjenu prema izrazu:

$$\hat{\psi}_P = \frac{\sum_{k=1}^{n} \frac{\hat{\psi}_k}{\Sigma_{\delta\rho_k}}}{\sum_{k=1}^{n} \frac{1}{\Sigma_{\delta\rho_k}}}. \tag{4.23}$$

Neka je

$$\delta\psi_k = \hat{\psi}_k - \hat{\psi}_P \tag{4.24}$$

udaljenost k-te točke mjerenja i procjene centra rotacijske nesigurnosti. Iterativno rješenje za kut α dano je prema $\alpha = \hat{\alpha} + \delta\alpha$:

$$\delta\alpha = \frac{\sum_{k=1}^{n} \frac{\delta\rho_k \delta\psi_k}{\Sigma_{\delta\rho_k}}}{\sum_{k=1}^{n} \frac{(\delta\psi_k)^2}{\Sigma_{\delta\rho_k}}}, \tag{4.25}$$

gdje su $\delta\rho_k$, $\delta\psi_k$ i $\Sigma_{\delta\rho_k}$ dani izrazima (4.19), (4.24) i (4.21).

Optimalni linijski segment određuje se iz dobivene linije projekcijom rubnih točaka, koje pripadaju početnoj procjeni linijskog segmenta, na optimalnu liniju. Jednostavnosti radi, zanemarujemo nesigurnost krajnjih točaka linijskih segmenata i matricu kovarijanci linijskog segmenta poistovjećujemo s matricom kovarijanci linija.

4.2.5 Procjena matrice kovarijanci linija

Neka je procjenjena orijentacija linije definirana kao zbroj stvarne orijentacije i pogreške:

$$\hat{\alpha} = \alpha + \epsilon_\alpha. \tag{4.26}$$

Slično, neka je udaljenost linije od ishodišta (duljina vektora normale) definirana kao:

$$\hat{\rho} = \rho + \epsilon_\rho. \tag{4.27}$$

Za ϵ_ρ i ϵ_α pretpostavimo da su slučajne varijable Gaussove raspodijele, srednje vrijednosti nula i pripadnih varijanci σ_ρ^2 i σ_α^2. Matrica kovarijanci linije definirana je kao:

$$\Sigma_L = \mathrm{E}\left[(\epsilon_L)(\epsilon_L)^T\right]^T, \tag{4.28}$$

gdje je E operator očekivanja, a $\epsilon_L = \begin{bmatrix} \epsilon_\rho & \epsilon_\alpha \end{bmatrix}^T$. Tada je matrica kovarijanci linije definirana sljedećim izrazom:

$$\Sigma_L = \begin{bmatrix} \mathrm{E}\left[\epsilon_\alpha^2\right] & \mathrm{E}\left[\epsilon_\alpha \epsilon_\rho\right] \\ \mathrm{E}\left[\epsilon_\alpha \epsilon_\rho\right] & \mathrm{E}\left[\epsilon_\rho^2\right] \end{bmatrix} = \begin{bmatrix} \Sigma_{\rho\rho} & \Sigma_{\rho\alpha} \\ \Sigma_{\rho\alpha} & \Sigma_{\alpha\alpha} \end{bmatrix}, \tag{4.29}$$

gdje su $\Sigma_{\rho\rho}$ varijanca udaljenosti linije od ishodišta, $\Sigma_{\alpha\alpha}$ varijanca orijentacije linije, dok su $\Sigma_{\rho\alpha}$ i $\Sigma_{\alpha\rho}$ kovarijance. Kako je po definiciji matrica kovarijanci simetrična pozitivno definitna matrica slijedi: $\Sigma_{\rho\alpha} = \Sigma_{\alpha\rho}$.

Prema otežanoj metodi pronalaženja linija matrica kovarijanci linije određuje se prema sljedećim izrazima:

$$\Sigma_{\rho\rho} = \frac{1}{\sum_{k=1}^{n} \frac{1}{\Sigma_{\delta\rho_k}}}, \tag{4.30}$$

$$\Sigma_{\alpha\alpha} = \frac{1}{\sum_{k=1}^{n} \frac{(\delta\psi_k)^2}{\Sigma_{\delta\rho_k}}}, \tag{4.31}$$

$$\Sigma_{\rho\alpha} = -\Sigma_{\alpha\alpha}\Sigma_{\rho\rho} \sum_{k=1}^{N} \frac{\delta\psi_k}{\Sigma_{\delta\rho_k}}, \tag{4.32}$$

gdje su $\delta\rho_k$, $\delta\psi_k$ i $\Sigma_{\delta\rho_k}$ dani izrazima (4.19), (4.24) i (4.21).

Nesigurnost odometrije

Definirajmo položaj robota u odnosu na globalni koordinatni sustav q_0:

$$q_i = \begin{bmatrix} x_i \\ y_i \\ \vartheta_i \end{bmatrix}, \tag{4.33}$$

čime je određen lokalni koordinatni sustav robota. Par (x_i, y_i) određuje poziciju, a ϑ_i orijentaciju robota u odnosu na pozitivni smjer x-osi. Sustav odometrije procjenjuje relativni pomak robota, integrirajući brzine kotača (kinematički model robota opisan je u potpoglavlju 5.1). Ako se robot giba od položaja q_i do položaja q_j definiran je lokalni

pomak q_{ij}:

$$q_{ij} = \begin{bmatrix} x_{ij} \\ y_{ij} \\ \vartheta_{ij} \end{bmatrix}. \tag{4.34}$$

Matrica kovarijanci pomaka dana je sljedećim izrazom:

$$\Sigma_{q_{ij}} = \begin{bmatrix} \Sigma_{xx} & \Sigma_{xy} & \Sigma_{x\vartheta} \\ \Sigma_{yx} & \Sigma_{yy} & \Sigma_{y\vartheta} \\ \Sigma_{\vartheta x} & \Sigma_{\vartheta y} & \Sigma_{\vartheta\vartheta} \end{bmatrix}. \tag{4.35}$$

Ovo je opći oblik matrice kovarijanci. Izgled matrice kovarijanci ovisi o primijenjenom modelu odometrije. Uz pretpostavku da su, za mali pomak q_{ij}, šumovi od x, y i ϑ nezavisne slučajne varijable, slijedi:

$$\Sigma_{q_{ij}} = \begin{bmatrix} \sigma_x^2 & 0 & 0 \\ 0 & \sigma_y^2 & 0 \\ 0 & 0 & \sigma_\vartheta^2 \end{bmatrix}, \tag{4.36}$$

gdje su σ_x^2, σ_y^2 i σ_ϑ^2 pripadne varijance pozicije x i y te orijentacije ϑ.

Neka je dan inicijalni položaj q_i, pripadna matrica kovarijanci toga položaja Σ_{q_i} te lokalni pomak q_{ij} s pripadnom matricom kovarijanci $\Sigma_{g_{ij}}$. Tada je kombinirana matrica kovarijance Σ_{q_j} u globalnom koordinatnom sustavu dana sljedećim izrazom:

$$\Sigma_{q_j} = Q\Sigma_{q_i}Q^T + K\Sigma_{q_{ij}}K^T, \tag{4.37}$$

gdje su:

$$Q = \begin{bmatrix} 1 & 0 & -y\cos\vartheta_i - x\sin\vartheta_i \\ 0 & 1 & -y\sin\vartheta_i + x\cos\vartheta_i \\ 0 & 0 & 1 \end{bmatrix} \tag{4.38}$$

i

$$K = \begin{bmatrix} \cos\vartheta_i & -\sin\vartheta_i & 0 \\ \sin\vartheta_i & \cos\vartheta_i & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{4.39}$$

Transformacija matrice kovarijanci linije u globalni koordinatni sustav

Neka je lokalni koordinatni sustav u i-tom položaju određen svojom pozicijom i orijentacijom u odnosu na globalni koordinatni sustav:

$$q_i = \begin{bmatrix} x_i \\ y_i \\ \vartheta_i \end{bmatrix}. \tag{4.40}$$

Ista se linija može promatrati iz različitih koordinatnih sustava, slika 4.15.

Razmotrimo liniju izmjerenu u i-tom lokalnom koordinatnom sustavu, gdje je $L_i = \begin{bmatrix} \alpha_i & \rho_i \end{bmatrix}^T$ i transformirajmo ju u globalni koordinatni sustav gdje ćemo ju označavati sa L_0:

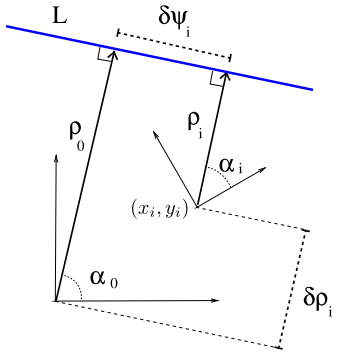
$$L_0 = \begin{bmatrix} \alpha_0 \\ \rho_0 \end{bmatrix} = \begin{bmatrix} \alpha_i + \vartheta_i \\ \rho_i + \delta\rho_i \end{bmatrix}, \tag{4.41}$$

gdje su $\delta\rho_i$ i $\delta\psi_i$ dani izrazima:

$$\delta\rho_i = x_i \cos(\alpha_i + \vartheta_i) + y_i \sin(\alpha_i + \vartheta_i), \tag{4.42}$$

$$\delta\psi_i = y_i \cos(\alpha_i + \vartheta_i) - x_i \sin(\alpha_i + \vartheta_i) \tag{4.43}$$

i označeni na slici 4.15.



Slika 4.15. *Linija u odnosu na globalni i lokalni koordinatni sustav. Vrijednosti $\delta\rho$ i $\delta\psi$ predstavljaju pomak koordinatnog sustava u ρ-ψ koordinatnom sustavu.*

Matricu kovarijanci Σ_{L_i} linije L_i u lokalnom koordinatnom sustavu q_i može se transformirati u globalni koordinatni sustav prema sljedećim izrazima:

$$\hat{\Sigma}_{L_0} = H \Sigma_{L_1} H^T, \tag{4.44}$$

gdje je:

$$H = \begin{bmatrix} 1 & 0 \\ \delta\psi_i & 1 \end{bmatrix}. \tag{4.45}$$

Važno je napomenuti da se svojstvene vrijednosti matrice ne mijenjaju s promjenom koordinatnog sustava.

Nadalje, linija L_i prikazana je u odnosu na lokalni koordinatni sustav q_i, gdje i sam položaj koordinatnog sustava ima nesigurnost zbog lokalizacije robota (koja može biti samo pomoću odometrije ili pomoću odometrije i lasera). Nesigurnost položaja koordinatnog sustava prikazana je matricom Σ_{q_i}. Neka je $\hat{\Sigma}_{q_0}$ doprinos nesigurnosti položaja ukupnoj nesigurnosti linije u globalnom koordinatnom sustavu. Da bismo izračunali $\hat{\Sigma}_{q_0}$, prvo zarotiramo Σ_{q_i} u kut linije α_0 u globalnom koordinatnom sustavu i zatim prebacimo nesigurnost položaja u nesigurnost linije, kao što je prikazano na slici

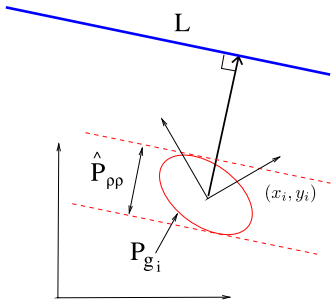
4.16 i prema sljedećem izrazu:

$$\hat{\Sigma}_{q_0} = K_i \Sigma_{q_i} K_i^T, \tag{4.46}$$

gdje je:

$$K_i = \begin{bmatrix} 0 & 0 & 1 \\ \cos(\alpha_i + \vartheta_i) & \sin(\alpha_i + \vartheta_i) & 0 \end{bmatrix}, \tag{4.47}$$

a matrica nesigurnosti položaja Σ_{q_i} opisana prema izrazu (4.37). Konačna matrica



Slika 4.16. *Projekcija nesigurnosti položaja u nesigurnost linije.*

kovarijanci linije je zbroj transformirane matrice kovarijanci linije i matrice kovarijanci položaja:

$$\Sigma_{L_0} = H_i \Sigma_{L_i} H_i^T + K_i \Sigma_{q_i} K_i^T. \tag{4.48}$$

Centar rotacijske nesigurnosti linije

Jednadžba (4.45) definira matricu transformacije matrice kovarijanci linije iz lokalnog koordinatnog sustava q_i u globalni q_0. Za svaku matricu kovarijanci Σ_L postoji posebna matrica H_P definirana izrazom:

$$H_P = \begin{bmatrix} 1 & 0 \\ \delta\psi_P & 1 \end{bmatrix}, \tag{4.49}$$

gdje je:

$$\delta\psi_P = -\frac{\Sigma_{\rho\alpha}}{\Sigma_{\alpha\alpha}}. \tag{4.50}$$

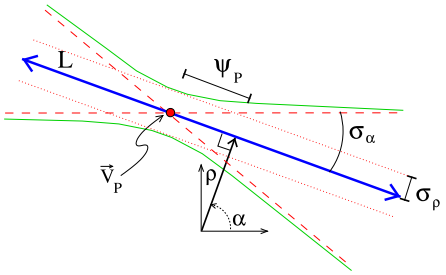
Matrica H_P dijagonalizira matricu kovarijanci Σ_L na sljedeći način:

$$\Sigma_L = \begin{bmatrix} 1 & 0 \\ \delta\psi_P & 1 \end{bmatrix} \begin{bmatrix} \sigma_\alpha^2 & 0 \\ 0 & \sigma_\rho^2 \end{bmatrix} \begin{bmatrix} 1 & \delta\psi_P \\ 0 & 1 \end{bmatrix}. \tag{4.51}$$

Točka V_P zove se *centar rotacijske nesigurnosti* u kojoj su ϵ_α i ϵ_ρ nezavisne slučajne varijable:

$$V_P = \begin{bmatrix} x_P \\ y_P \end{bmatrix} = \begin{bmatrix} \rho\cos(\alpha) - \delta\psi_P \sin(\alpha) \\ \rho\sin(\alpha) + \delta\psi_P \cos(\alpha) \end{bmatrix}. \tag{4.52}$$

Na slici 4.17 prikazana je linija s pripadnim nesigurnostima. Crtkane linije predstavljaju nesigurnost kuta, a točkaste linije dio prikazuju pripadnu nesigurnost udaljenosti linije od ishodišta. U sjecištu crtkanih linija nalazi se centar rotacijske nesigurnosti V_P.



Slika 4.17. *Linija s označenim centrom rotacijske nesigurnosti i pripadnim nesigurnostima.*

4.2.6 Spajanje sličnih linijskih segmenata

χ^2-test

Do sada smo mjerne podatke lasera opisivali linijama. Međutim, nismo provjeravali slaganje pretpostavljenog modela linije s podacima, tj. točnosti pretpostavljenog modela. U formulaciji testiranja hipoteza, želimo razlučiti između dvije hipoteze: H0 – model je točan i H1 – model je netočan. Moguće je testirati nultu hipotezu H0 za danu razinu značajnosti p.

χ^2-test uobičajena je metoda za testiranje ispravnosti hipoteze. Test se sastoji od izračuna slučajne varijable χ^2_n koja ima χ^2 razdiobu, određenu prema mjerenjima i modelu koji se testira. Za danu n-dimenzionalnu slučajnu varijablu V i mjerenje te varijable $\hat{V}$ mora vrijediti pretpostavka da V ima normalnu razdiobu, srednje vrijednosti 0 i matrice kovarijanci Σ_V. Vrijednost slučajne varijable χ^2_n može se računati prema:

$$\chi^2_n = (V - \hat{V})^T (\Sigma_V)^{-1} (V - \hat{V}). \tag{4.53}$$

Obično se za danu razinu značajnosti p definira jednosmjerno područje odbijanja hipoteze

$$\chi^2_n > c, \tag{4.54}$$

gdje konstanta c zadovoljava uvjet

$$\int_c^\infty p(\chi^2_n) d\chi^2_n = p, \tag{4.55}$$

gdje je $p(\chi^2_n)$ gustoća χ^2 razdiobe reda n. Za dane n i p, vrijednost c određuje se numerički ili očitavanjem iz χ^2 tablice razdiobe vjerojatnosti.

U radu je izabrana velika razina značajnosti, jer se ne želi odbaciti niti jedna hipoteza

koja bi mogla biti istinita. Važno je napomenuti da velika vrijednost χ^2_n implicira malu vjerojatnost odbacivanja istinitih hipoteza. Sam test ne implicira veliku vjerojatnost da je model točan. On, naprotiv, samo služi kao učinkovita metoda za odbacivanje netočnih hipoteza.

Spajanje sličnih linija χ^2-testom

Hipoteza koja se provjerava jest pripadnost dvaju linijskih segmenata pripadaju istoj liniji. Spajanje sličnih linija zasnivano je na najrobusnijim parametrima linijskog segmenta, orijentaciji α i duljini vektora normale ρ.

Promatramo dva linijska segmenta S_0^i i S_0^j, njihove linije L_0^i i L_0^j te pripadne matrice kovarijanci linija u globalnom koordinatnom sustavu. Zatim zbrajamo matrice kovarijance da izračunamo ukupnu relativnu nesigurnost razlike između dviju linija:

$$\Sigma_{\delta L} = \Sigma_{L_0^i} + \Sigma_{L_0^j}. \tag{4.56}$$

Razlika između dviju linija računa se prema:

$$\delta L = \begin{bmatrix} \delta\alpha \\ \delta\rho \end{bmatrix} = \begin{bmatrix} \alpha_0^i - \alpha_0^j \\ \rho_0^i - \rho_0^j \end{bmatrix}. \tag{4.57}$$

χ^2-test je oblika:

$$\chi_2^2 = (\delta L)^T (\Sigma_{\delta L})^{-1} (\delta L) < 3. \tag{4.58}$$

Ovaj je test dovoljan za linijske segmente koji imaju vrlo malu nesigurnost orijentacije.

Da bismo mogli koristiti linije s većom rotacijskom nesigurnosti, nelinearnosti uvedene međusobnom ovisnosti nesigurnosti parametara α i ρ moraju se umanjiti. To se može napraviti tako da linije uspoređujemo u koordinatnom sustavu gdje je međusobna ovisnost najmanja. Centar rotacijske nesigurnosti definiran je prema (4.52) kao točka u kojoj se minimiziraju efekti ovisnosti parametara α i ρ. Obje se linije transformiraju u koordinatni sustav s ishodištem u centru rotacijske nesigurnosti za kombiniranu nesigurnost (zbroj) obaju linija. Linije L_0^i i L_0^j i matrica kovarijance $\Sigma_{\delta L}$ transformiraju se u $\tilde{L}_0^i$, $\tilde{L}_0^j$ i $\tilde{\Sigma}_{\delta L}$. Tada se izvodi sljedeći test:

$$\delta\tilde{L} = \begin{bmatrix} \delta\tilde{\alpha} \\ \delta\tilde{\rho} \end{bmatrix} = \begin{bmatrix} \tilde{\alpha}_0^i - \tilde{\alpha}_0^j \\ \tilde{\rho}_0^i - \tilde{\rho}_0^j \end{bmatrix}, \tag{4.59}$$

$$\tilde{\chi}_2^2 = (\delta\tilde{L})^T (\tilde{\Sigma}_{\delta L})^{-1} (\delta\tilde{L}) < 3. \tag{4.60}$$

Ako je uvjet (4.60) ispunjen može se odrediti s odabranom razinom značajnosti p da se razlika linija L_0^i i L_0^j može opravdati modeliranim šumom te hipoteza nije odbačena.

Primjenom formulacije maksimalne vjerodostojnosti dobivaju se izrazi za spojenu

liniju i njezinu nesigurnost:

$$\Sigma_{Ls} = ((\Sigma_{L_0^i})^{-1} + (\Sigma_{L_0^j})^{-1})^{-1}, \tag{4.61}$$

$$L_s = \Sigma_{Ls}((\Sigma_{L_0^i})^{-1} L_0^i + (\Sigma_{L_0^j})^{-1} L_0^j). \tag{4.62}$$

Krajnje se točke dobivaju projekcijom postojećih krajnih točaka na novo dobivenu liniju. Kako smo zbog jednostavnosti zanemarili nesigurnost krajnjih točaka, dva segmenta koja se nalaze na istoj liniji, a krajnje su im točke udaljene za manje od $3\sigma_d$, smatramo jednim linijskim segmentom.

4.3 Algoritam istraživanja nepoznatog prostora zasnivan na skočnim bridovima i izgradnji linijske karte

U ovom je potpoglavlju opisan je razvijeni algoritam istraživanja prostora koji je proširenje Ekmanova algoritma. Proširenje uključuje promjenu kriterijske funkcije i uključenje utjecaja mjernog šuma očitanja senzora i nesigurnosti položaja.

4.3.1 Kriterij odabira sljedeće mjerne pozicije

Razvijeni algoritam istraživanja koristi izmijenjeni kriterij izbora najboljeg skočnog brida [53], [54]. Ekmanov algoritam koristi kriterij u obliku odnosa dobitka iz čvora n_j i cijene putanje od čvora n_i do n_j. Već malo manji iznos dobitka čvora, može se promatrati kao da je duljina putanje do tog čvora toliko puta veća. Prema tome, može se odabrati dalji čvor, koji ima veći iznos dobitka, umjesto nekog bližeg čvora s manjim iznosom dobitka. Budući da se svi čvorovi iz grafa istraživanja trebaju posjetiti, to će značiti da se robot mora kasnije vraćati na čvorove s malim iznosom dobitka. U unutarnjim prostorima s mnogo vrata, stolova i uskih prolaza, ta se situacija događa dosta često, što uzrokuje nezadovoljavajuće predugo trajanje procesa istraživanja.

Za ubrzanje procesa istraživanja u radu je primijenjena kriterijska funkcija koja umjesto odnosa dobitaka koristi zbroj dobitka čvora i cijene putanje normirane prema maksimalnom dometu senzora. Pri tome se kao iznos dobitka čvora koristi duljina skočnog brida, budući da ona upućuje na veličinu neistraženog prostora. Kriterij za izbor najboljeg skočnog brida definiran je kao:

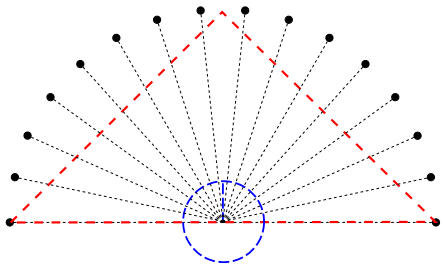
$$C(n_i, n_j) = \frac{|h_j|}{R_v} + \frac{R_v}{c_{ij}}, \tag{4.63}$$

gdje je h_j skočni brid ispred mjernog položaja n_j, R_v polumjer vidljivosti (maksimalan domet senzora), a c_{ij} cijena putanje od čvora n_i do čvora n_j. Ovaj oblik kriterija podjednako otežava dulje bridove i kraću putanju do potencijalne mjerne pozicije.

4.3.2 Određivanje poligona mjerenja iz linijskih segmenata

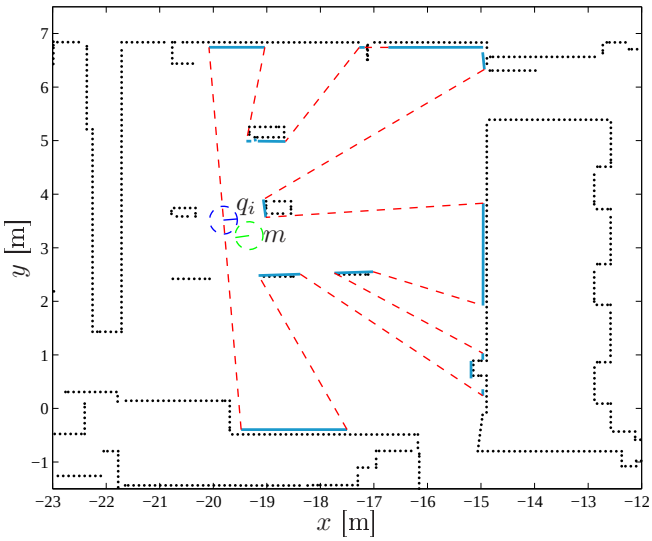
U Ekmanovu algoritmu istraživanja poligon mjerenja P_i stvara se ekspanzijom i fuzijom zvjezdastog poligona dobivenog iz mjernih podataka senzora udaljenosti. Prošireni poligon mjerenja EP_i stvara se iz poligona mjerenja P_i tako što se proširuje skočnim bridovima. Prošireni poligon istraživanja $EP_E(i)$ dobiva se iz proširenog poligona istraživanja iz prethodnog mjerenja $EP_E(i-1)$ i novog proširenog poligona mjerenja EP_i. Dobiveni skočni bridovi definiraju nove mjerne pozicije za neistraženi prostor. Tijekom fuzije pokušavaju se otkriti kolinearni bridovi, koji se zamjenjuju jednim većim bridom. Međutim, kolinearni bridovi se često ne mogu otkriti zbog šuma te se dobiva velik broj nepotrebnih bridova.

Razvijeni algoritam istraživanja koristi postupak izdvajanja linijskih segmenata koji pronalazi statistički točne bridove iz zašumljenih očitanja senzora i uzima u obzir nesigurnost položaja mjerenja. Dobiveni linijski segmenti zatvaraju se u poligon tako da se vrhovi linijskih segmenata poredaju u lokalnom koordinatnom sustavu po kutu u intervalu od $[-\pi, \pi]$. Tako poredani vrhovi $s_1, \ldots, s_n$ definiraju prošireni poligon mjerenja. Posebno, ako pozicija robota p_i nije kolinearna sa $\overline{s_n s_1}$, poligon se zatvara preko pozicije robota, odnosno umjesto $\overline{s_n s_1}$ nastaju $\overline{s_n p_i}$ $\overline{p_i s_1}$. Linijski segmenti odgovaraju l bridovima proširenog poligona mjerenja, a bridovi koji su potrebni za zatvaranje poligona odgovaraju h bridovima proširenog poligona mjerenja EP_i. Zbog jasnoće uvodimo oznaku $\widetilde{EP}_i$, pod kojom se podrazumijeva dobiveni poligon iz linijskih segmenata. Ako nema dobivenih linijskih segmenata iz mjerne pozicije, skočni se bridovi definiraju tako da čine trokut upisan u polukružnicu određenu polumjerom vidljivosti laserskog senzora udaljenosti. Jedan skočni brid je promjer polukružnice, a druga dva brida spajaju krajnje točke promjera s točkom na polukružnici prema kojoj je robot orijentiran, kao što je prikazano na slici 4.18.



Slika 4.18. *Dobivanje skočnih bridova u slučaju u kojem nema očitanih prepreka.*

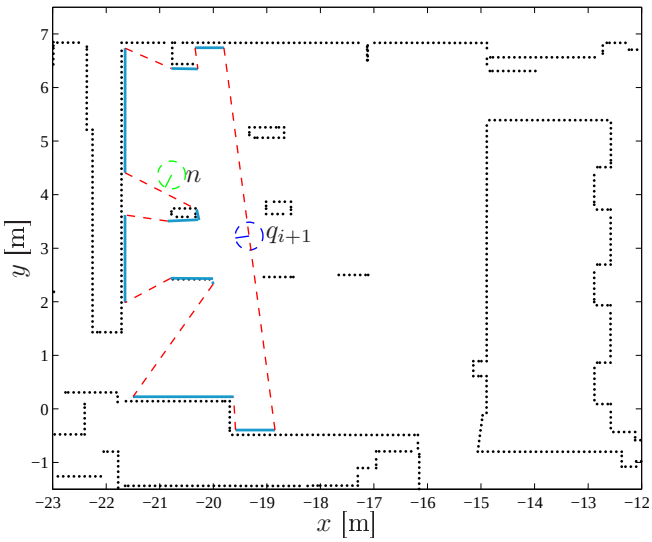
Na slici 4.19 prikazan je primjer mjerenja dijela nepoznatog prostora. Stvarna karta koja je robotu nepoznata prikazana je točkama. Crtkani bridovi (crveni) su skočni bridovi, a puni bridovi (plavi) su linijski segmenti izdvojeni iz mjerenja lasera. Položaj označen s q_i je trenutni položaj robota, a položaj označen s m je sljedeći odabrani mjerni položaj. Sljedeći mjerni položaj odabran je ispred skočnog brida, da bi se osiguralo nadovezivanje poligona, odnosno njihovo djelomično preklapanje. Skočni bridovi



Slika 4.19. *Primjer određivanja skočnih bridova (crtkane linije) i stvarnih bridova (pune linije) iz mjerenja laserskog senzora iz položaja q_i.*

zajedno s linijskim segmentima čine prošireni poligon mjerenja $\widetilde{EP}_i$.

Slika 4.20 nadovezuje se na sliku 4.19 i prikazuje mjerenje iz odabranog sljedećeg mjernog položaja q_{i+1}. Trenutni mjerni položaj označen je s q_{i+1}, a odabrani sljedeći

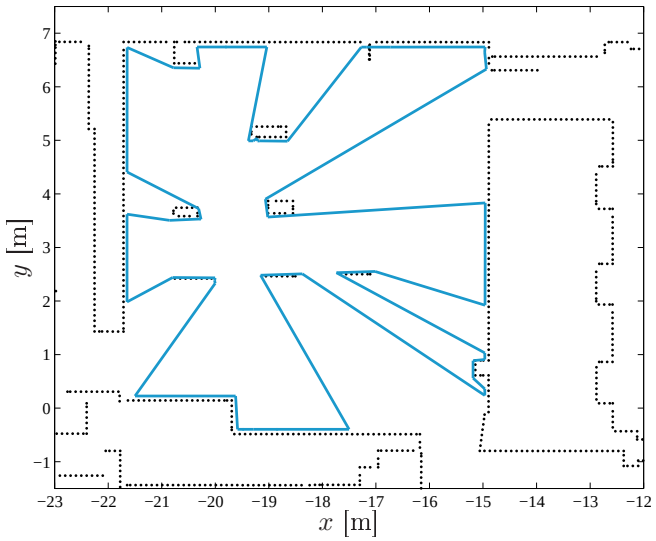


Slika 4.20. *Određivanje skočnih bridova (crtkane linije) i linijskih segmenata (pune linije) iz odabranog sljedećeg mjernog položaja q_{i+1}.*

mjerni položaj s n. Skočni bridovi zajedno s linijskim segmentima čine prošireni poligon mjerenja $\widetilde{EP}_{i+1}$.

U Ekmanovu se algoritmu istraživanja smanjuje broj skočnih bridova u proširenom poligonu istraživanja ponovnom ekspanzijom i fuzijom, a ovdje se koristi operacija unije

poligona prema postupku općenitog isijecanja poligona [133] nad proširenim poligonom mjerenja $\widetilde{EP}_i$ i proširenim poligonom istraživanja iz prethodnog mjerenja $\widetilde{EP}_E(i-1)$. Unija poligona izvodi se tako da se od dva poligona sačuvaju samo rubni bridovi (najudaljeniji). Na taj se način definira istraženi prostor te sprječava stvaranje skočnih bridova unutar istraženog prostora. Skočni bridovi, čija je duljina veća od širine robota, smatraju se kandidatima za novo mjerenje. Primjer proširenog poligona istraživanja $\widetilde{EP}_E(i+1)$, dobivenog unijom poligona $\widetilde{EP}_i$ i $\widetilde{EP}_{i+1}$ sa slika 4.19 i 4.20, prikazan je na slici 4.21. Pseudokod razvijenog algoritma istraživanja dan je algoritmom 4.2.



Slika 4.21. *Poligon dobiven unijom poligona i pripadni skočni bridovi nakon unije poligona.*

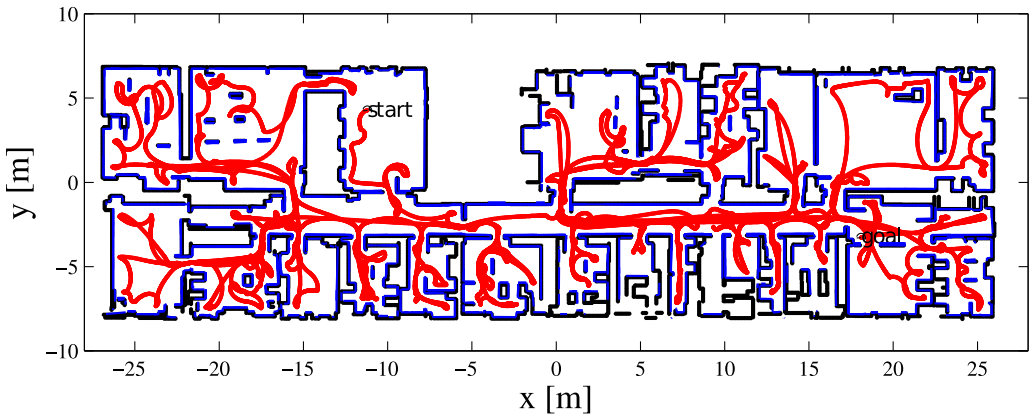
4.4 Eksperimentalni rezultati

Ekmanov algoritam i razvijeni algoritam istraživanja testirani su u slimulatoru i eksperimentalno na stvarnom robotu. Simulacija je izvedena u karti prostora čitavog kata Zavoda za automatiku i računalno inženjerstvo. Odabrana je lokalizacija zanemarive nesigurnosti. Laserski je senzor udaljenosti u simulaciji gotovo idealan te su time osigurani uvjeti za izvođenje Ekmanova algoritma. I Ekmanov i razvijeni algoritam daju podjednake rezultate koji su prikazani na slici 4.22, gdje je prikazana trajektorija robota. Oba su algoritma u potpunosti istražili čitav prostor koji je dostupan robotu. Dobiveni poligon istraživanja prikazan je na slici 4.23. Rezultati simulacije gotovo u potpunosti odgovaraju zadanoj karti zbog zanemarivih nesigurnosti lokalizacije i laserskog senzora udaljenosti.

Eksperimenti su izvršeni na stvarnom robotu na dijelu ograđenog prostora Zavoda za automatiku i računalno inženjerstvo. Za provjeru uspješnosti algoritma istraživanja korištena je karta prostora prikazana točkama, koja robotu nije poznata. Valja napomenuti da ta karta ne odgovara u potpunosti stvarnom prostoru. Primjerice, u karti su sva

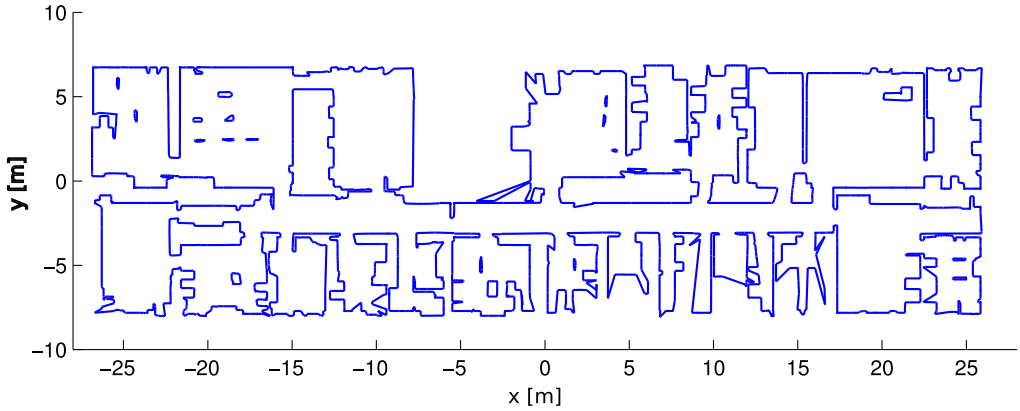
Algoritam 4.2: Razvijeni algoritam istraživanja

$n_0 \leftarrow (0,0)\{Inicijalizacija\}$
$n_1 \leftarrow (0,0)$
$\mathcal{G}_E(0) \leftarrow ((0,0), \emptyset)$
$i \leftarrow 1$
ponavljaj
  idi-na-novu-mjernu-poziciju(n_i)
  $\mathcal{R}_i \leftarrow$ **novo-mjerenje**
  $\{S_i\} \leftarrow$ **izdvoji-linijske-segmente**$(\mathcal{R}_i)$
  ako $|\{S_i\}| > 0$
    $\{S_i^0\} \leftarrow$ **spoji-slične-linijske-segmente**$(\{S_i\}, \{S^0\})$
    $\widetilde{EL_i} \leftarrow$ **poredaj-bridove-po-kutu**$(\{S_i^0\})$
    $\widetilde{EP_i} \leftarrow$ **nadopuni-bridove-u-poligon**(EL_i)
  inače
    $\widetilde{EP_i} \leftarrow$ **skočni-bridovi-za-maksimalni-domet**$(\mathcal{R}_i)$
  kraj
  $\widetilde{EP_E}(i) \leftarrow \widetilde{EP_E}(i-1) \cup \widetilde{EP_i}$
  $\mathcal{G}_E(i) \leftarrow$ **stvori-graf-istraživanja**$(\mathcal{G}_E(i-1), \widetilde{EP_E}(i))$
  $n_{i+1} \leftarrow$ **odaberi-čvor**$(n_i, \mathcal{G}_E(i))$
  $i \leftarrow i+1$
dok nije $g_i = 0$

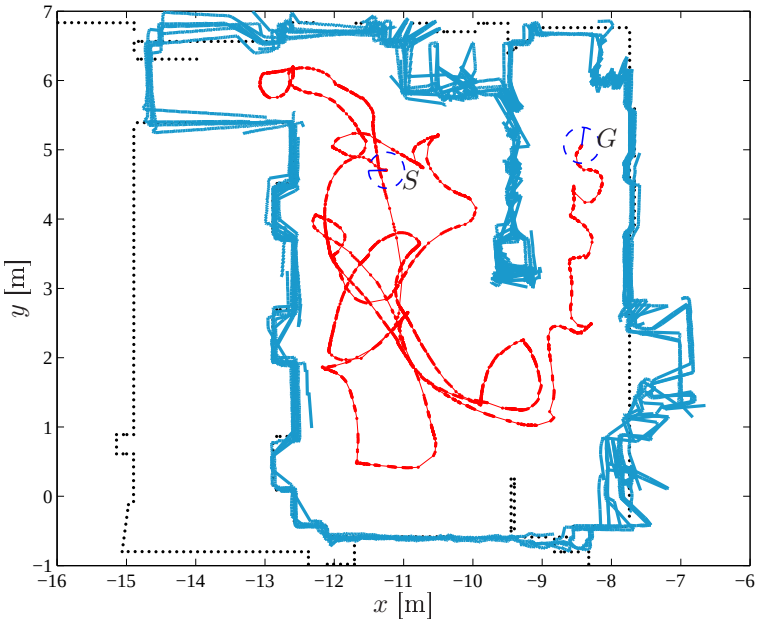


Slika 4.22. *Rezultati simulacije razvijenog algoritma istraživanja prostora.*

vrata prikazana kao otvorena, a u stvarnosti su zatvorena, a stube su zagrađene ravnom linijom. Također, neke prepreke nisu zabilježene, kao što su stol, biljke i stolci kojima su zagrađene stube iz sigurnosnih razloga. Robot je lokaliziran AMCL lokalizacijom (engl. *adaptive Monte Carlo localization*), koja koristi opisanu kartu. Gibanje robota između mjernih položaja izvođeno je algoritmom planiranja gibanja koji objedinjuje planiranje putanje i izbjegavanje prepreka (opisan u poglavlju 5). Na slici 4.24 prikazani su eksperimentalni rezultati Ekmanova algoritma istraživanja prostora [58]. Prikazana je trajektorija robota za vrijeme istraživanja i snimljena karta u obliku bridova. Robot



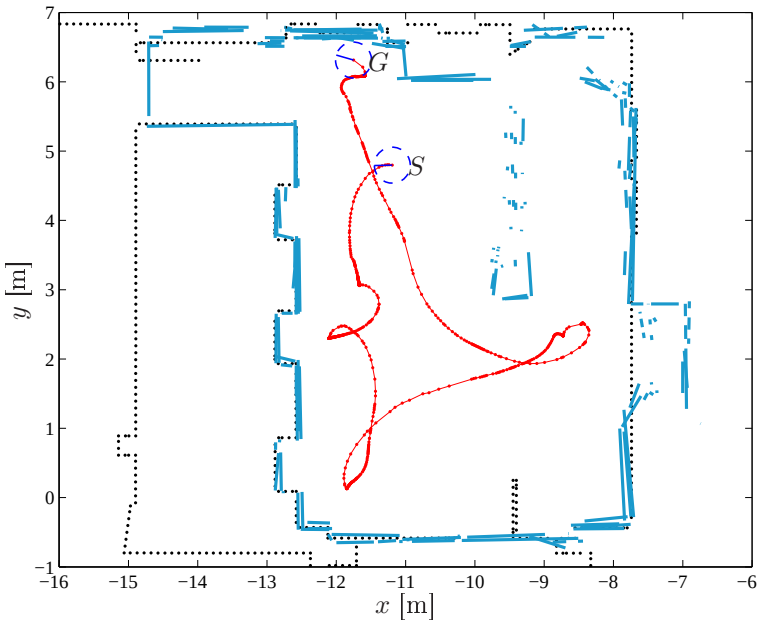
Slika 4.23. *Dobiveni poligon nakon završenog simulacijskog eksperimenta.*



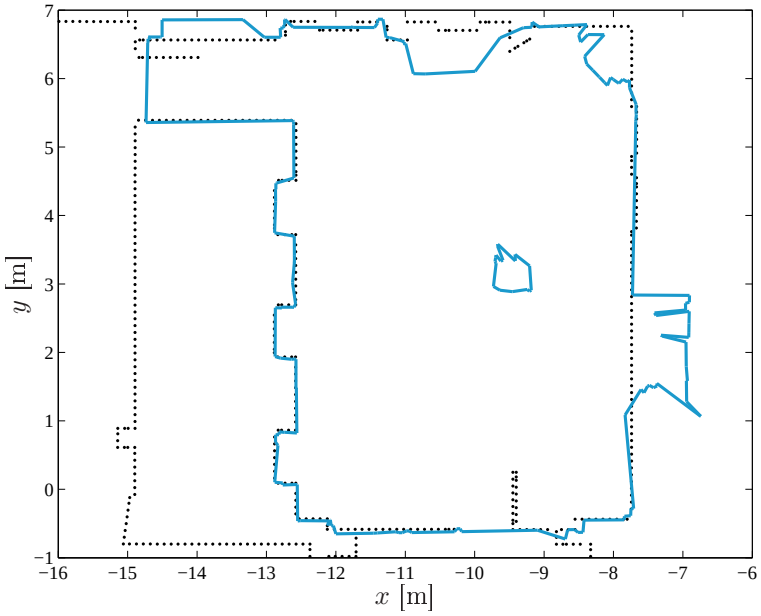
Slika 4.24. *Eksperimentalni rezultati Ekmanova algoritma istraživanja prostora.*

je snimao kartu iz 62 mjerne pozicije pri čemu je snimio 11547 linijskih segmenata koji predstavljaju kartu istražene prostorije.

Na slici 4.25 prikazani su eksperimentalni rezultati razvijenog algoritma istraživanja. Prikazana je trajektorija robota za vrijeme istraživanja i snimljena karta u obliku bridova. Rezultati pokazuju samo 13 mjernih pozicija te 162 linijska segmenta. Dobiveni poligon prikazan je na slici 4.26.



Slika 4.25. *Eksperimentalni rezultati razvijenog algoritma istraživanja prostora.*



Slika 4.26. *Dobiveni poligon nakon završenog eksperimenta.*

4.5 Sažetak

U ovom je poglavlju razmatrano autonomno istraživanje nepoznatog unutarnjeg poligonalnog prostora. Prikazan je razvijeni algoritam istraživanja poligonalnih prostora, kao proširenje Ekmanova algoritma istraživanja. Razvijenim su algoritmom istraživanja

uklonjena stroga ograničenja Ekmanova algoritma koji pretpostavlja idealnu lokalizaciju robota i idealni senzor udaljenosti. Dok kod Ekmanova algoritma mjerni podaci senzora udaljenosti predstavljaju uzorkovan *poligon vidljivosti*, razvijeni algoritam uključuje algoritam izdvajanja linijskih segmenata prema Pfisteru *et al.*, kojim se pronalaze statistički točni bridovi poligona vidljivosti iz zašumljenih očitanja senzora, uzimajući pritom u obzir utjecaj nesigurnosti položaja mobilnog robota. U Ekmanovu se algoritmu bridovi poligona vidljivosti, koji ne odgovaraju bridovima prostora, nazivaju *skočnim bridovima*, a strategija se istraživanja zasniva na činjenici da skočni bridovi navode na smjerove vjerojatno neistraženih područja prostora. U razvijenom se algoritmu računaju aproksimacije skočnih bridova dobivene zatvaranjem linijskih segmenata u poligon. Kombiniranjem podataka senzora udaljenosti iz različitih mjernih položaja, postupno se izgrađuje karta, u kojoj skočni bridovi dijele istražena i neistražena područja. Dok je kod Ekmanove metode pretpostavka da se poligoni vidljivosti iz dvaju susjednih mjernih položaja savršeno poklapaju ili nadovezuju, u razvijenom se algoritmu primijenjuje statistička metoda slaganja bridova iz dvaju mjernih položaja. Kod Ekmanova algoritma savršenim preklapanjem ili nadovezivanjem poligona vidljivosti nestaju skočni bridovi koji se nalaze u već istraženom području. Nasuprot tomu, u razvijenom se algoritmu koristi postupak unije dvaju poligona za eliminiranje skočnih bridova unutar već istraženog prostora.

U Ekmanovu je algoritmu pokazano da je prostor istražen kada u karti više ne postoji niti jedan skočni brid. Razvijeni algoritam nasljeđuje to svojstvo, a stvorena karta prostora objedinjuje utjecaj nesigurnosti senzorskih očitanja i položaja mobilnog robota, čime je postignuta konvergencija algoritma istraživanja u realnim uvjetima. Eksperimentalni rezultati na stvarnom robotu potvrđuju prednosti razvijenog algoritma istraživanja nad Ekmanovim algoritmom.

Poglavlje 5

Slijeđenje putanje i izbjegavanje prepreka

Putanja dobivena algoritmom pretraživanja grafa sastavljenog iz mrežaste karte zauzeća u prostoru mobilnog robota je geometrijska krivulja sastavljena od ravnih segmenata s oštrim kutevima na prijelazu između dvaju segmenata. Takva putanja nije prikladna za slijeđenje zbog kinematičkih i dinamičkih ograničenja robota. Naime, praćenje ravnih segmenata kojima se smjer naglo mijenja mora rezultirati znatnim usporavanjem gibanja mobilnoga robota. Međutim, po dijelovima prekinuta geometrijska putanja može se pretvoriti u neki oblik glatke krivulje koja u vremenu predstavlja trajektoriju mobilnog robota na osnovi koje se robotu zadaju referentne brzine gibanja. Razlikuju se neposredni i posredni pristupi generiranja trajektorije.

U neposrednim se pristupima generiranja trajektorije, primjerice kubični splajn [103], [102] ili interpolacija gradijentnom funkcijom [65], moraju uzeti u obzir dinamička i kinematička ograničenja robota. Dobivena krivulja prostorno odstupa od izvorne geometrijske putanje, pa se mora dodatno provjeriti prolazi li ona kroz neku prepreku u prostoru. U slučaju dinamičkih prostora, reparametriranje globalne trajektorije ovakvim metodama računski je dosta zahtjevno.

Posredni pristupi generiranja trajektorije zasnivaju se na primjeni algoritama izbjegavanja prepreka. Najčešće korišteni algoritmi izbjegavanja prepreka mogu se podijeliti u dvije skupine: (1) algoritmi smjernog pristupa i (2) algoritmi brzinskog pristupa. Algoritmi smjernog pristupa traže optimalni smjer u kojem se robot treba gibati. Primjeri algoritama smjernog pristupa su: metoda umjetnog potencijalnog polja [59], metoda histograma vektorskog polja [10] i algoritam dijagrama blizine [82]. Ovi algoritmi ne uzimaju u obzir dinamička ograničenja robota, za razliku od algoritama brzinskog pristupa. U brzinskom pristupu pretražuje se prostor translacijskih i rotacijskih brzina robota. Traži se optimalan par brzina (translacijska i rotacijska brzina) prema nekoj kriterijskoj funkciji koja uzima u obzir lokalnu konfiguraciju prepreka i usmjerenost prema cilju. Tipična pretpostavka kod algoritama ovog pristupa jest da se robot giba po kružnim lukovima. Primjeri algoritama brzinskog pristupa su: metoda krivulje-brzine [121],

metoda trake-krivulje [122] i algoritam dinamičkog prozora [36].

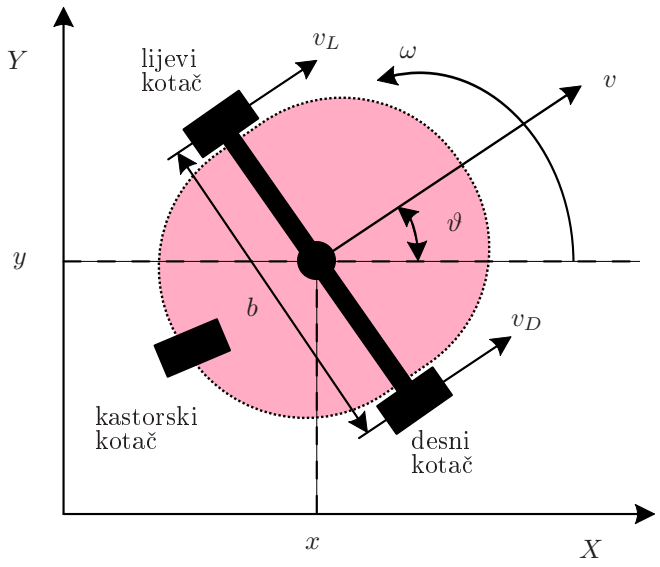
Algoritmi izbjegavanja prepreka podložni su problemu lokalnog minimuma. Lokalni minimum je pozicija koja nije cilj, a u kojoj se robot trajno zaustavi. Najčešći test pojave lokalnog minimuma jest postavljanje prepreke u obliku slova "U" između robota i cilja, tako da ga gibanje prema cilju približava prepreci i njome je okružen s tri strane. Odmicanje od prepreke također znači i odmicanje od cilja i tako robot trajno ostaje u istoj točki. Rješenje je ovoga problema integracija algoritma izbjegavanja prepreka i algoritma planiranja putanje.

Postoje brojni pristupi koji kombiniraju lokalnu metodu izbjegavanja prepreka s globalnom metodom planiranja putanje [13], [2], [5]. Međutim, ti pristupi nekad generiraju gibanje koje nije optimalno. Primjera radi, zamislimo dva međusobno okomita hodnika koja tvore slovo "T". Zadatak je robota prijeći iz jednog hodnika u drugi. Zbog neplaniranog profila brzina duž cijele trajektorije, robot može propustiti skretanje u drugi hodnik jer mu dinamička ograničenja brane da ostvari proizvoljno nagli zaokret. Stoga mora zaokrenuti i probati ponovo s druge strane istog hodnika. Iz drugog pokušaja uspjeva skrenuti, jer mu se brzina smanjila, no moguće je zamisliti scenarij u kojem robot trajno oscilira oko ulaza u drugi hodnik. Glavni je razlog ovog problema što se sljedeća upravljačka akcija odabire s obzirom na trenutno stanje robota i globalnu isplaniranu putanju (samo x, y koordinate), a u određenim situacijama, posebno pri velikim brzinama robota, nužno je planirati i brzine robota. U [125] je opisan uspješni pristup zasnovan na algoritmu dinamičkog prozora, koji planira gibanje u petero-dimenzionalnom prostoru položaja i brzina $(x, y, \vartheta, v, \omega)$ u neposrednoj blizini robota. U [86] je opisan pristup zasnovan na algoritmu dinamičkog prozora kojem je dokazana konvergencija prema cilju. U nastavku će biti detaljnije opisan algoritam dinamičkog prozora i predložena metoda objedinjavanja algoritma dinamičkog prozora i algoritma FD*.

5.1 Kinematički model mobilnog robota s diferencijalnim pogonom

U radu je korišten mobilni robot s diferencijalnim pogonom. Sastoji se od tri kotača. Dva su kotača pogonska čijim se kutnim brzinama može neovisno upravljati. Treći je kotač kastorski kotač koji se može slobodno okretati, a služi za stabilizaciju gibanja robota. Oba pogonska kotača sadrže enkodere za mjerenje brzina ili prijeđenih putova kotača. Pomoću brzina kotača moguće je izračunati pomak robota u prostoru. Odometrija je postupak određivanja trenutnog položaja mobilnog robota pomoću nekog prethodno određenog položaja i poznatih brzina (mjerenih "odometrom") u danom vremenu [9]. Ovdje su izdvojena tri postupka odometrije prema [120] od kojih jedan koristimo za određivanje trajektorija.

Neka je $q = [x \ y \ \vartheta]^T$ položaj mobilnog robota, gdje su (x, y) Kartezijeve koordinate središta osovine kotača robota, a ϑ je orijentacija glavne osi robota s obzirom na pozitivnu poluos X (slika 5.1). Neka je $\mathcal{Q}$ skup svih položaja koje mobilni robot može



Slika 5.1. *Kinematički model mobilnog robota s diferencijalnim pogonom.*

poprimiti u prostoru. Pretpostavljamo da je $\mathcal{Q}$ kontinuirani i ograničeni podskup od $\mathbb{R}^3$.

Opisani mobilni robot može se gibati (uz isključeno klizanje) samo u smjeru svoje uzdužne osi. Prema tome sljedeće kinematičko ograničenje mora biti zadovoljeno:

$$\dot{x}\sin(\vartheta) - \dot{y}\cos(\vartheta) = 0. \tag{5.1}$$

Ovo ograničenje svrstava mobilnog robota s diferencijalnim pogonom u neholonomske sustave. Neholonomski sustav je onaj u kojemu su kinematička ograničenja neintegrabilna i ne mogu se pretvoriti u geometrijska ograničenja [81]. Geometrijska ograničenja mobilnog robota uvode prepreke u prostoru. Dakle, kinematičko ograničenje ne utječe na geometrijsko ograničenje mobilnog robota. Robot može doći do bilo koje točke u slobodnom prostoru, primjerice, okretanjem u mjestu pa gibanjem ravno naprijed. Drugim riječima u neholonomskim sustavima gibanje je ograničeno lokalno, ali ne i globalno. Neholonomski sustavi nazivaju se još i sustavi s neholonomskim ograničenjima.

Kinematički model podrazumijeva samo gibanje robota bez proklizavanja kotača, ne uračunava ubrzanja/usporenja i sve promjene brzina smatra trenutnima. Kinematički model mobilnog robota primijenjujući neholonomsko ograničenje (5.1) opisan je sljedećim jednadžbama:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\vartheta} \end{bmatrix} = \begin{bmatrix} \cos\vartheta \\ \sin\vartheta \\ 0 \end{bmatrix} v + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \omega, \tag{5.2}$$

gdje su translacijska brzina v i rotacijska brzina ω dane sljedećim relacijama:

$$\begin{aligned} v &= \frac{v_D + v_L}{2} = \frac{\omega_D k_R + \omega_L k_R}{2} \\ \omega &= \frac{v_D - v_L}{b} = \frac{\omega_D k_R - \omega_L k_R}{b}, \end{aligned} \tag{5.3}$$

gdje je k_R polumjer kotača, b duljina pogonske osovine, v_L i v_D obodne brzine lijevog i desnog kotača, a ω_L i ω_D kutne brzine lijevog i desnog kotača. Iako se robotom upravlja preko kutnih brzina lijevog i desnog kotača, veličine v i ω smatraju se ulaznim veličinama. Uvodimo pretpostavku da su brzine v i ω konstantne unutar svakog vremenskog intervala (ciklusa). Ta je pretpostavka općenito zadovoljena u implementacijama digitalnog sustava upravljanja. Stoga slijedi da se za vrijeme k-tog intervala robot giba po kružnom luku polumjera $r = \dfrac{v(k)}{\omega(k)}$ koji se preinačuje u linijski segment za $\omega(k) = 0$. Pretpostavimo da je položaj robota $q(k)$ u trenutku k poznat, zajedno s vrijednostima upravljačkih veličina $v(k)$ i $\omega(k)$ primijenjenim u intervalu $[t(k), t(k+1))$. Položaj robota $q(k+1)$ u trenutku $k+1$ može se rekonstruirati Eulerovom metodom numeričke integracije kinematičkog modela (5.2):

$$\begin{aligned} x(k+1) &= x(k) + v(k)\Delta t \cos\vartheta(k) \\ y(k+1) &= y(k) + v(k)\Delta t \sin\vartheta(k) \\ \vartheta(k+1) &= \vartheta(k) + \omega(k)\Delta t, \end{aligned} \tag{5.4}$$

gdje je $\Delta t = t(k+1) - t(k)$ duljina trajanja vremenskog intervala, odnosno period diskretizacije. Pogreška u određivanju $x(k+1)$ i $y(k+1)$ jest u tome što se uzima konstantan iznos $\vartheta(k)$ u cijelom intervalu Δt. Ovi su izrazi egzaktni za linijske segmente ($\omega(k) = 0$). Često se koristi i estimacija kinematičkog modela, dobivena drugim redom integracijske metode Runge-Kutta:

$$\begin{aligned} x(k+1) &= x(k) + v(k)\Delta t \cos(\vartheta(k) + \frac{\omega(k)\Delta t}{2}) \\ y(k+1) &= y(k) + v(k)\Delta t \sin(\vartheta(k) + \frac{\omega(k)\Delta t}{2}) \\ \vartheta(k+1) &= \vartheta(k) + \omega(k)\Delta t, \end{aligned} \tag{5.5}$$

Prva dva izraza u (5.5) koriste srednju vrijednost orijentacije unutar intervala Δt. Numerička pogreška u ovim formulama je manja nego u izrazu (5.4).

Položaj robota $q(k+1)$ u trenutku $k+1$ može se rekonstruirati egzaktnom integra-

cijom kinematičkog modela (5.2):

$$\begin{aligned} x(k+1) &= x(k) + \frac{v(k)}{\omega(k)}\left(\sin\vartheta(k+1) - \sin\vartheta(k)\right) \\ y(k+1) &= y(k) - \frac{v(k)}{\omega(k)}\left(\cos\vartheta(k+1) - \cos\vartheta(k)\right) \\ \vartheta(k+1) &= \vartheta(k) + \omega(k)\Delta t. \end{aligned} \tag{5.6}$$

Ovi izrazi egzaktno opisuju kružni luk i bit će korišteni u opisu trajektorija robota. Za $\omega(k) = 0$ prva dva izraza u (5.6) su i dalje definirane i jednake onima u (5.4) i (5.5) (lako se pokazuje korištenjem izraza $\lim_{x\to 0}\frac{\sin x}{x} = 1$). Međutim, u implementaciji je potrebno posebno razdvojiti izraze na dva slučaja.

Kinematički se model može zapisati u obliku prijelazne jednadžbe stanja

$$q(k+1) = f(q(k), u(k)), \tag{5.7}$$

gdje je upravljačka vrijednost $u(k) = [v(k)\ \omega(k)]^T$. Pretpostavimo da robot trenutno postiže zadane upravljačke vrijednosti. U trenutku $k+1$, prije nego što je primijenjena nova upravljačka vrijednost $u(k+1)$ vrijedit će stara vrijednost $u(k)$:

$$\begin{aligned} v_R(k+1) &= v(k) \\ \omega_R(k+1) &= \omega(k), \end{aligned} \tag{5.8}$$

gdje s $v_R(k+1)$ i $\omega_R(k+1)$ označavamo translacijsku i rotacijsku brzinu robota u trenutku $k+1$. Položaj $q(k)$ zajedno s brzinama $v_R(k)$ i $\omega_R(k)$ nazivamo stanjem robota, $s(k) = [q(k)\ v_R(k)\ \omega_R(k)]^T$.

Upravljačke vrijednosti posjeduju kinematička i dinamička ograničenja. Kinematička ograničenja određena su najmanjim i najvećim brzinama (translacijskim i rotacijskim) koje robot može ostvariti. Stoga za $u(k)$ mora vrijediti sljedeće:

$$\begin{aligned} 0 &\leq v(k) \leq v_{max}, \\ -\omega_{max} &\leq \omega(k) \leq \omega_{max}, \end{aligned} \tag{5.9}$$

gdje su v_{max} i ω_{max} pozitivne realne vrijednosti koje odgovaraju najvećoj translacijskog brzini (reverzne brzine su zabranjene) i najvećoj rotacijskoj brzini (robot i u lijevu i u desnu stranu može okretati jednako brzo). Dinamička ograničenja određuju najveće promjene brzine u jednom vremenskom intervalu koje robot može ostvariti. Stoga za $u(k)$ s obizrom na $u(k-1)$ vrijedi sljedeće:

$$\begin{aligned} -\Delta v_{max} &\leq v(k) - v(k-1) \leq \Delta v_{max}, \\ -\Delta\omega_{max} &\leq \omega(k) - \omega(k-1) \leq \Delta\omega_{max}, \end{aligned} \tag{5.10}$$

gdje su Δv_{max} i $\Delta\omega_{max}$ pozitivne realne vrijednosti koje odgovaraju najvećoj promjeni translacijske brzine i najvećoj promjeni rotacijske brzine u intervalu Δt, odnosno naj-

većim primijenjenim ubrzanjima/usporenjima u intervalu Δt.

Neka je υ *upravljačka sekvenca*, a τ *trajektorija robota* definirane kako slijedi:

$$\begin{aligned} &\upsilon = \{u(0), u(1), \ldots, u(F-1)\}, \\ &\tau = \{q(0), q(1), \ldots, q(F)\}, \\ &\text{uz } q(k+1) = f(q(k), u(k)),\ k = 0, \ldots, F-1, \end{aligned} \tag{5.11}$$

gdje je $q(0)$ početni položaj robota. Svako stanje $q(k)$ na trajektoriji τ određeno je prema prijelaznoj jednadžbi (5.7) iz prethodnog stanja $q(k-1)$ i vrijednosti $u(k-1)$ u upravljačkoj sekvenci, računajući redom za $k = 1, \ldots, F$.

5.2 Slijeđenje putanje zasnovano na dinamičkom prozoru

5.2.1 Algoritam dinamičkog prozora

Algoritam dinamičkog prozora jest reaktivna metoda izbjegavanja prepreka u kojoj se optimalna upravljačka vrijednost $u = [v\ \omega]^T$ traži u prostoru brzina koje robot može ostvariti u sljedećem vremenskom intervalu s obzirom na njegova kinematička i dinamička ograničenja. Trajektorija robota predstavljena je nizom kružnih lukova i ravnih linija [36] uz pretpostavku da su brzine robota konstantne unutar vremenskog intervala $[t_i, t_{i+1}]$.

Početni skup brzina $\mathcal{V}_{kin}$ uključuje sve moguće upravljačke vrijednosti s obzirom na kinematička ograničenja robota prema (5.9):

$$\mathcal{V}_{kin} = \left\{ [v\ \omega]^T \mid v \in [0, v_{max}] \wedge \omega \in [-\omega_{max}, \omega_{max}] \right\}. \tag{5.12}$$

Skup brzina $\mathcal{V}_{din}$ uključuje sve moguće upravljačke vrijednosti s obzirom na dinamička ograničenja robota prema (5.10). Taj se skup određuje prema trenutnoj brzini robota u i-tom vremenskom trenutku $v_R(i)$ i $\omega_R(i)$, koja je jednaka primijenjenoj upravljačkoj vrijednosti u $(i-1)$-vom vremenskom trenutku, $v_R(i) = v(i-1)$ i $\omega_R(i) = \omega(i-1)$:

$$\mathcal{V}_{din} = \left\{ [v\ \omega]^T \mid \begin{array}{c} v \in [v_R(i) - \Delta v_{max}, v_R(i) + \Delta v_{max}] \wedge \\ \omega \in [\omega_R(i) - \Delta\omega_{max}, \omega_R(i) + \Delta\omega_{max}] \end{array} \right\}. \tag{5.13}$$

Upravljačka vrijednost $[v\ \omega]^T$ smatra se dopuštenom ako se robot nakon primijenjene upravljačke vrijednosti u stigne zaustaviti, poštujući ograničenja (5.10), prije nego udari u prepreku. Skup dopuštenih brzina $\mathcal{V}_{dop}$ određuje se prema:

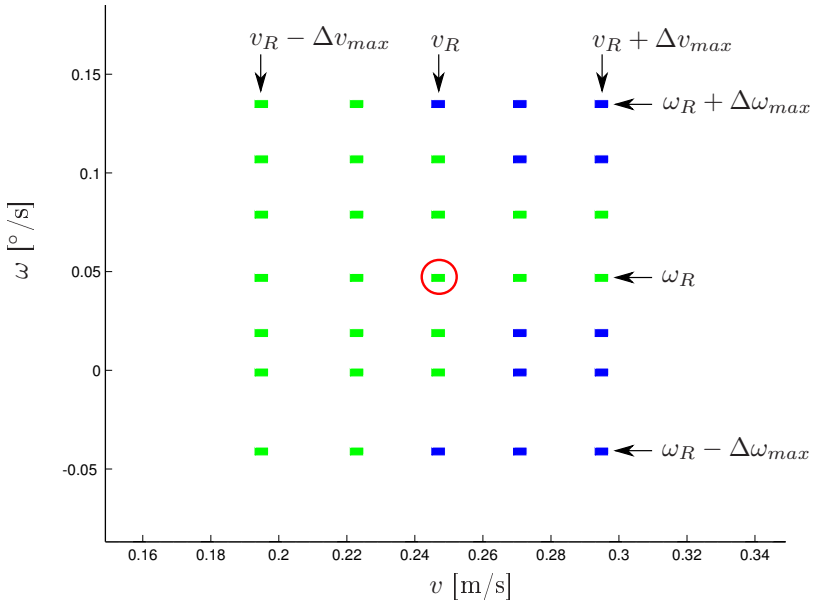
$$\mathcal{V}_{dop} = \left\{ [v\ \omega]^T \mid v \leq \sqrt{2\rho_{min}(v,\omega)a_{max}} \wedge \omega \leq \sqrt{2\varphi_{min}(v,\omega)\alpha_{max}} \right\}, \tag{5.14}$$

gdje $\rho_{min}(v,\omega)$ i $\varphi_{min}(v,\omega)$ predstavljaju udaljenost i kut do najbliže prepreke po odgovarajućoj krivulji, a $a_{max} = \dfrac{\Delta v_{max}}{\Delta t}$ i $\alpha_{max} = \dfrac{\Delta\omega_{max}}{\Delta t}$ su najveće translacijsko i rotacijsko ubrzanje/usporenje.

Skup koji čini rezultantni prostor pretraživanja dobiven je presjekom navedenih triju skupova brzina:

$$\mathcal{V} = \mathcal{V}_{kin} \cap \mathcal{V}_{din} \cap \mathcal{V}_{dop}. \tag{5.15}$$

Pretpostavka je da robot može ostvariti bilo koju realnu vrijednost brzine unutar opisanih ograničenja, prema tome $\mathcal{V} \subset \mathbb{R}^2$. Međutim, da bi pretraživanje bilo ostvarivo u stvarnom vremenu mora se napraviti diskretizacija skupa $\mathcal{V}$ tako da se novi prostor pretraživanja $\mathcal{V}_d$ sastoji od konačnog broja upravljačkih vrijednosti. Primjer diskretiziranog prostora brzina $\mathcal{V}_d$ prikazan je na slici 5.2.



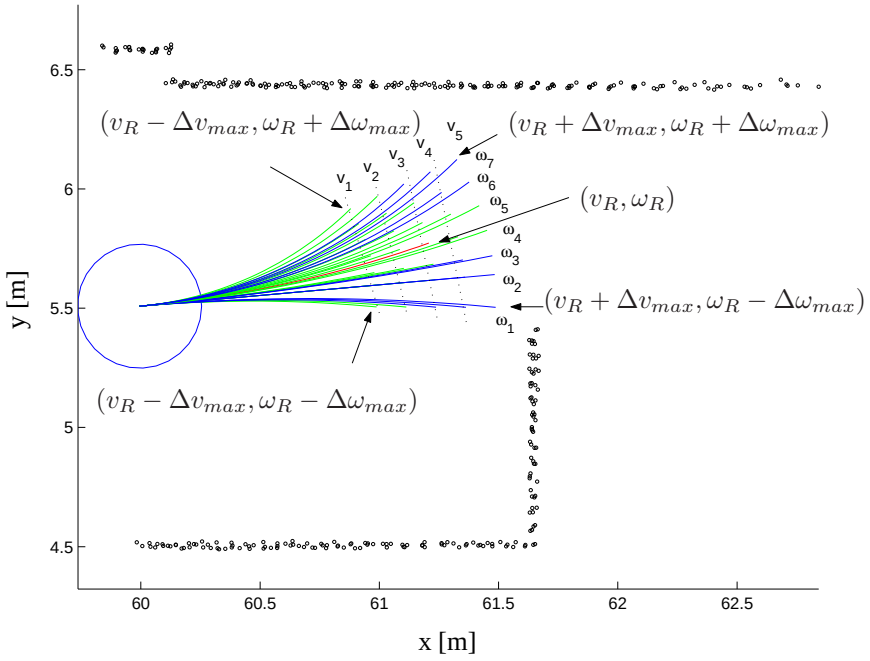
Slika 5.2. *Diskretizirani prostor brzina.*

Optimalna upravljačka vrijednost $u^* \in \mathcal{V}_d$ je ona koja maksimizira određenu kriterijsku funkciju Γ. Da bi se odredila trajektorija iz položaja robota $q(i)$ u trenutku i do danog cilja u sljedećih T vremenskih intervala, moraju se odrediti vrijednosti $u_i(k) = [v_i(k)\ \omega_i(k)]^T$ koje se moraju izvršiti u svim vremenskim intervalima $k = 0, \ldots, T-1$. Tada prostor pretraživanja raste eksponencijalano u donosu na broj razmotrenih intervala. Da bi pretraživanje bilo ostvarivo i prikladno za rad u stvarnom vremenu, algoritam dinamičkog prozora uzima u obzir samo prvi interval za odabir optimalnog para brzina i pretpostavlja da brzine ostaju konstantne u preostalih $T-1$ interval. Time se pretražuje dvodimenzionalan diskretan prostor brzina $\mathcal{V}_d$, što je ostvarivo u polinomnom vremenu. Pretraživanje se izvršava svaki korak (vremenski interval), a optimalna upravljačka vrijednost $u_i^*(0) = [v_i^*(0)\ \omega_i^*(0)]^T$ prosljeđuje se kao naredba izvršnom članu. Svaka upravljačka vrijednost $u \in \mathcal{V}_d$ jednoznačno određuje kružnu trajektoriju dinamičkog prozora τ_i^u u sljedećih T vremenskih intervala s obzirom na trenutni položaj robota

$q(i)$:

$$\begin{aligned} &\tau_i^u = \{q_i^u(0), q_i^u(1), \ldots, q_i^u(T)\} \\ &\text{uz } q_i^u(k+1) = f(q_i^u(k), u),\ k = 0, \ldots, T-1. \end{aligned} \tag{5.16}$$

gdje je $q_i^u(0) = q(i)$, a indeks i označuje vremenski trenutak proračuna trajektorije u sljedećih T vremenskih intervala. Primjer trajektorija dobivenih iz skupa $\mathcal{V}_d$ prikazanog na slici 5.2 prikazan je na slici 5.3.



Slika 5.3. *Trajektorije robota dobivene iz diskretiziranog prostora brzina sa slike 5.2.*

Kriterijska funkcija Γ izražena je kao otežani zbroj triju kriterijskih funkcija [36]:

$$\Gamma(v, \omega) = \alpha_1\ prohodnost(v, \omega) + \alpha_2\ usmjerenje(v, \omega) + \alpha_3\ brizna(v, \omega). \tag{5.17}$$

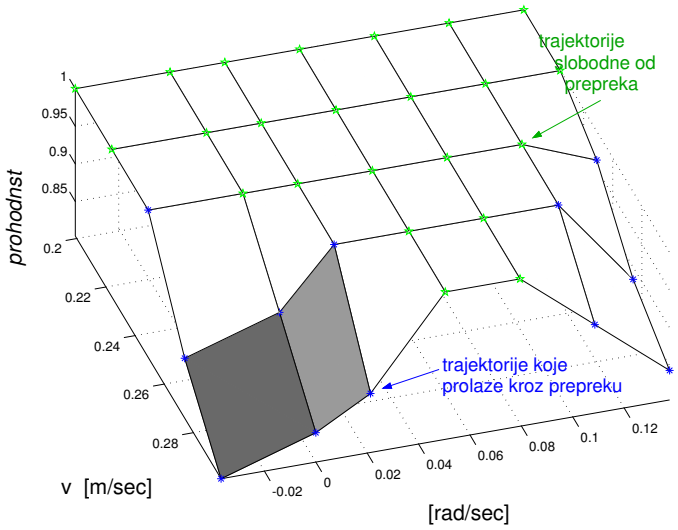
Kriterijska funkcija *prohodnost* daje veću vrijednost trajektorijama koje su dalje od prepreka. Konfiguracija prepreka u okolini robota uključena je u ovu kriterijsku funkciju tako što se po svakoj kružnoj trajektoriji računa očekivani prevaljeni put ρ_{min} od robota do najbliže prepreke. Prepreke u lokalnoj okolini robota predstavljene su skupom točaka koje je izmjerio senzor udaljenosti. U [97] se umjesto udaljenosti računa vrijeme do sudara s preprekom t_{sudar}, što uzima u obzir brzinu trajektorije $u = [v\ \omega]^T$. Kriterijska funkcija definirana je kako slijedi:

$$prohodnost(v, \omega) = \begin{cases} 0 & \text{za } t_{sudar} \leq T_z \\ \dfrac{t_{sudar} - T_z}{T_{max} - T_z} & \text{za } T_z < t_{sudar} < T_{max}, \\ 1 & \text{za } t_{sudar} > T_{max} \end{cases} \tag{5.18}$$

gdje je $T_z(v,\omega) = \max\{\frac{v}{a_{max}}, \frac{\omega}{\alpha_{max}}\}$ vrijeme zaustavljanja po trajektoriji određenoj s $u = [v\ \omega]^T$, a $t_{sudar}(v,\omega) = \frac{\rho_{min}(v,\omega)}{v}$ vrijeme do sudara robota s najbližom preprekom. Točka sudara robota s preprekom jest točka na trajektoriji u kojoj robot svojom konturom dodiruje prepreku. Umjesto ovakvog određivanja točke sudara u [36] se koristi tzv. brzinski ovisna bočna prohodnost BP_v kao dodatno proširenje polumjera robota pa je točka sudara dodir proširene konture robota i prepreke. Bočna prohodnost BP_v raste linearno s v, a njen je utjecaj prividno suženje slobodnog prostora (hodnici, prolazi između prepreka) pri većim brzinama. T_{max} je dopušteno vrijeme do sudara i predstavlja gornju vremensku granicu iznad koje se trajektorije smatraju slobodnima od prepreka. T_{max} predstavlja vremenski trenutak zadnje točke trajektorije dinamičkog prozora, odnosno $T_{max} = T\Delta t$. Broj vremenskih intervala T odabire se prema korištenom dometu senzora S_{max} i najvećoj translacijskoj brzini na trajektoriji v_{max}

$$T = \lfloor \frac{S_{max}}{v_{max}\Delta t} \rfloor. \tag{5.19}$$

Prema tome, sve trajektorije dinamičkog prozora nalaze se unutar okoline koju robot može promatrati. Primjer kriterijske funkcije *prohodnost* koja odgovara trajektorijama i konfiguraciji prepreka na slici 5.3 prikazana je na slici 5.4.



Slika 5.4. *Kriterijska funkcija prohodnost*(v,ω).

Kriterijska funkcija *brzina* daje veću vrijednost većim translacijskim brzinama, ako

je robot daleko od cilja, a manju ako je blizu cilja:

$$brzina(v,\omega) = \begin{cases} \dfrac{v}{v_{max}} & \text{ako je robot daleko od cilja} \\ 1 - \dfrac{v}{v_{max}} & \text{ako je robot blizu cilja} \end{cases} \tag{5.20}$$

Kriterijska funkcija *usmjerenje* daje veću vrijednost trajektorijama u smjeru cilja. Neka je položaj robota $q_P = [x_P\ y_P\ \vartheta_P]^T$ određen gibanjem robota jedan vremenski interval po kružnom luku, određenom s $u = [v\ \omega]^T$, i nakon toga zaustavljanjem najvećim kutnim usporenjem. Neka je točka $P(x_P, y_P)$ pozicija robota u prostoru koja odgovara položaju q_P, a točka $G(x_G, y_G)$ cilj. Usmjerenje prema cilju određeno je kutom koji čine os robota u položaju q_P i vektor $\overrightarrow{GP}$:

$$usmjerenje(v,\omega) = \frac{1}{2} - \frac{1}{2}\cos(\vartheta_P - \overrightarrow{GP}). \tag{5.21}$$

Ovakvom definicijom usmjerenja prema cilju omogućeno je aperiodsko približavanje orijentacije robota željenoj orijentaciji. Definicija ovog kriterija samo preko inkrementalnog pomaka orijentacije u sljedećem koraku, bez zaustavljanja [111], uzrokuje oscilacije u postizanju željene orijentacije robota prema cilju. Kao što je izloženo u [32] i [36], ovaj je pristup podložan lokalnom minimumu. Lokalni se minimum može izbjeći uvođenjem povezanosti slobodnog prostora prema cilju. U algoritmu globalnog dinamičkog prozora [13] kriterijska funkcija ima četiri člana:

$$\Gamma(v,\omega) = \alpha_1\ nf1(v,\omega) + \alpha_2\ usmjerenje(v,\omega) + \alpha_3\ brzina(v,\omega) + \alpha_4\ \Delta NF1(v,\omega). \tag{5.22}$$

Navigacijska funkcija NF1 daje informaciju o globalnoj putanji [68]. Ova metoda propagira valnu frontu od cilja pridjeljujući svakoj točki mreže cijenu putanje do cilja. U bilo kojoj točki postoji informacija o gradijentu smanjenja udaljenosti do cilja.

Iako je u (5.22) riješen problem lokalnog minimuma, ostaje problem određivanja iznosa težinskih koeficijenata $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ koji značajno utječu na izvedbu gibanja robota.

Nasuprot navigacijskoj funkciji NF1, algoritmi pretraživanja grafova određuju jedinstvenu putanju od starta do cilja, što omogućuje je određivanje mjere slaganja trajektorija određenih dinamičkim prozorom i putanje. Algoritam reduciranog dinamičkog prozora [2] koristi lokalnu geometrijsku putanju kojoj unaprijed pridjeljuje profil translacijskih brzina. Time ubrzava izvođenje algoritma dinamičkog prozora (umjesto pretraživanja dvo-dimenzionalnog prostora brzina pretražuje se jedno-dimenzionalni prostor), ali ne osigurava glatko gibanje prema cilju. U ovome se radu predlaže metoda objedinjavanja algoritma FD* i algoritma dinamičkog prozora koja izbjegava spomenute nedostatke.

5.2.2 Objedinjavanje algoritma FD* i dinamičkog prozora

U ovome se radu predlaže nova kriterijska funkcija koja uspoređuje trajektorije robota dobivene algoritmom dinamičkog prozora s geometrijskom globalnom putanjom dobivenom algoritmom FD* na mrežastoj karti zauzeća. Kriterijska funkcija $\Gamma(v,\omega)$ izražena je kao otežani zbroj dviju kriterijskih funkcija: funkcije prohodnosti $prohodnost(v,\omega)$ i funkcije slaganja trajektorije s putanjom $putanja(v,\omega)$:

$$\Gamma(v,\omega) = \alpha\ prohodnost(v,\omega) + (1-\alpha)\ putanja(v,\omega), \tag{5.23}$$

gdje je α težinski koeficijent. Isti oblik kriterijske funkcije $\Gamma(v,\omega)$ koristili smo u [113], ali je funkcija slaganja trajektorije s putanjom bila drugačija.

Kriterijska funkcija $putanja(v,\omega)$ mjeri slaganje mogućih trajektorija robota s *efektivnom putanjom*. Efektivna je putanja dužina koja spaja trenutnu poziciju robota $R(x(i),y(i))$ i referentnu točku na globalnoj putanji $E(x_E,y_E)$, odnosno $\overline{RE}$. Orijentacija efektivne putanje određuje referentnu orijentaciju robota u odnosu na lokalnu konfiguraciju globalne putanje određujući tako referentnu rotacijsku brzinu robota ω_{ref}. Duljina efektivne putanje $|\overline{RE}|$ određuje referentnu udaljenost koju robot treba prijeći, određujući tako referentnu translacijsku brzinu robota v_{ref}. Efektivna je putanja parametrizirana u vremenu tako što je opisana s $T+1$ točkom međusobno ekvidistantnih razmaka i odgovarajućim vremenskim trenucima $0,\Delta t,\ldots,T\Delta t$. Referentna točka E ima pridružen vremenski trenutak $T_{max} = T\Delta t$. Efektivna trajektorija τ_i^E iz položaja robota $q(i)$ dana je s:

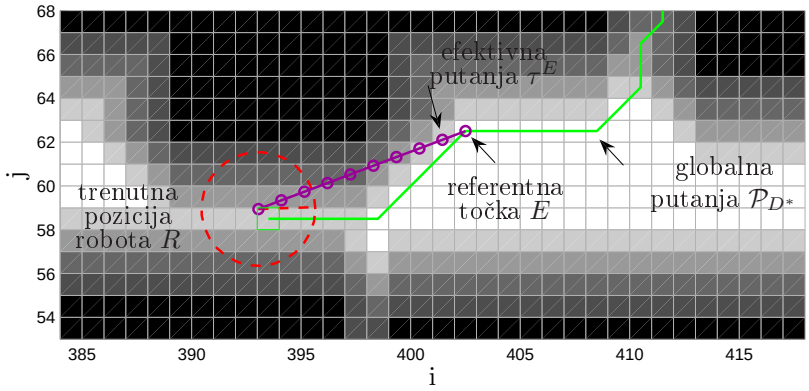
$$\begin{aligned} &\tau_i^E = \{q_i^E(0), q_i^E(1), \ldots, q_i^E(T)\} \\ &\text{uz } q_i^E(k+1) = f(q_i^E(k), u_i^E),\ u_i^E = [v_i^E\ 0]^T,\ k = 0,\ldots,T-1, \end{aligned} \tag{5.24}$$

gdje je: $q_i^E(0) = q(i)$, $v_i^E = \dfrac{|\overline{RE}|}{T_{max}}$ translacijska brzina efektivne trajektorije, a indeks i vremenski trenutak proračuna trajektorije u sljedećih T vremenskih intervala. Slaganje trajektorije dinamičkog prozora i efektivne putanje određeno je zbrajanjem udaljenosti točaka na obje trajektorije, kako slijedi:

$$putanja(v,\omega) = 1 - \frac{\sum_{k=0}^{T} \|(x_i^u(k), y_i^u(k)) - (x_i^E(k), y_i^E(k))\| - D_{min}}{D_{max} - D_{min}}, \tag{5.25}$$

gdje gornji indeks u označava trajektoriju dinamičkog prozora određenu s $u = [v\ \omega]^T$. Na taj je način određeno odstupanje trajektorije od globalne putanje. Granične vrijednosti D_{min} i D_{max} korištene su za normiranje kriterija u područje od 0 do 1 po svim mogućim trajektorijama.

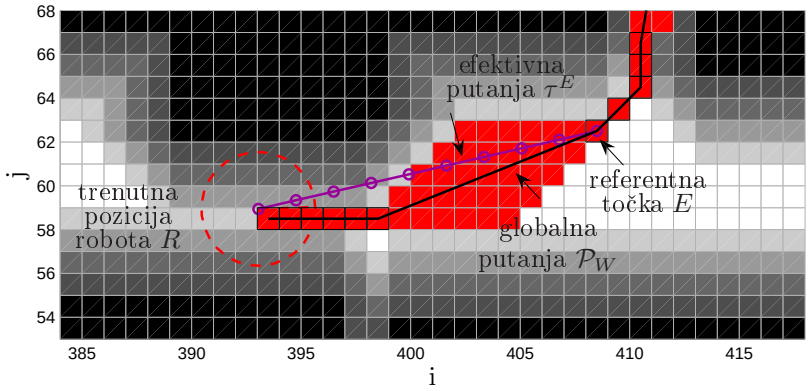
Budući da je globalna putanja sastavljena od ravnih segmenata promjenjive orijentacije, odabrali smo poziciju referentne točke E na drugoj promjeni orijentacije putanje, kao što je prikazano na slici 5.5. Taj je izbor zasnovan na činjenici da druga promjena orijentacije putanje određuje hoće li se smjer putanje vratiti na izvornu ori-



Slika 5.5. *Određivanje referentne točke E na globalnoj putanji $\mathcal{P}^*$.*

jentaciju ili će se nastaviti mijenjati, što posredno predstavlja mjeru zakrivljenosti putanje. Pozicija referentne točke E također je određena s obzirom na njenu udaljenost od trenutne pozicije robota R. Udaljenost referentne točke ograničena je odozgo s $d^E_{max}(v_R) = (v_R + \Delta v_{max})T_{max}$ tako da uzima u obzir najveću translacijsku brzinu koja se može primijeniti u sljedećem vremenskom intervalu Δt. Ako je druga promjena smjera putanje udaljenija od robota od duljine d^E_{max}, referentna se točka određuje kao točka na globalnoj putanji udaljena točno d^E_{max} od robota. Udaljenost referentne točke ograničena je odozdo s $d^E_{min} = \frac{1}{2}a_{max}T^2_{zM}$, čime se filtriraju beznačajne promjene putanje u blizini robota. Određena je s obzirom na najveće translacijsko ubrzanje a_{max} i najveće vrijeme zaustavljanja $T_{zM} = \frac{v_{max}}{a_{max}}$ tako da se robot može zaustaviti po najkraćoj efektivnoj putanji, ako je prethodno postigao najveću translacijsku brzinu.

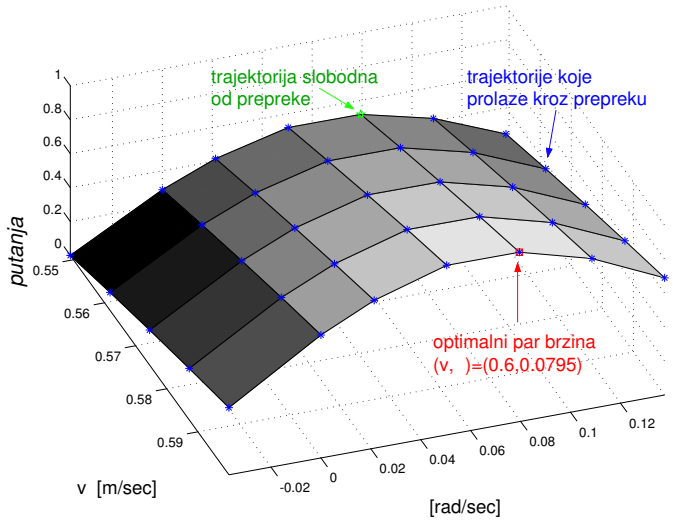
Ova se metoda slijeđenja putanje može primjeniti na bilo kojoj globalnoj geometrijskoj putanji ako se odredi odgovarajuća efektivna putanja. Korištenje najkraće putanje $\mathcal{P}_W$ određenom algoritmom DD* (potpoglavlje 2.4) za određivanje efektivne putanje prikazano je na slici 5.6. Iz slika 5.5 i 5.6 mogu se usporediti efektivne putanje s global-



Slika 5.6. *Određivanje referentne točke na globalnoj putanji $\mathcal{P}_W$.*

nim putanjama $\mathcal{P}_{D^*}$ i $\mathcal{P}_W$ pod istim uvjetima. Konfiguracija prepreka prikazana je u obliku mrežaste karte zauzeća sa sigurnosnom maskom cijena. Druga promjena smjera globalne putanje $\mathcal{P}_W$ mnogo je dalje nego druga promjena smjera globalne putanje $\mathcal{P}_{D^*}$. Ako je robot blizu prepreke, kriterij prohodnosti utječe na biranje nižih brzina i stoga je duljina d^E_{max} manja. Prema tome, duljina efektivne putanje duža je u slobodnom prostoru, a kraća među preprekama. Duža efektivna putanja znači grublju aproksimaciju geometrijske globalne putanje i veći odmak robota od globalne putanje.

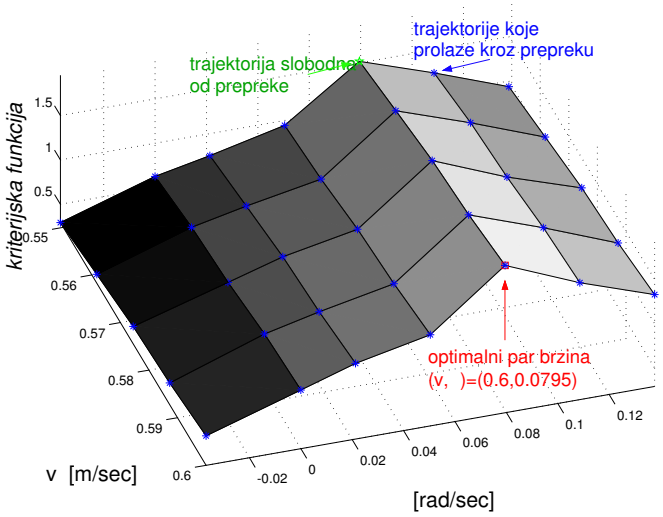
Kriterijska funkcija $putanja(v,\omega)$ prikazana je na slici 5.7, a globalna kriterijska funkcija za koeficijent $\alpha = 0.5$ prikazana je na slici 5.8. U ovome primjeru optimalna brzina maksimizira kriterijsku funkciju $putanja(v,\omega)$, ali ne maksimizira kriterijsku funkciju prohodnosti. Ako se koeficijent α odabere tako da je slobodna trajektorija optimalna ($\alpha = 0.9$ za ovaj slučaj), robot će usporiti (odabrat će se najmanja brzina iz $\mathcal{V}_d$, $0.55\frac{m}{s}$ umjesto $0.6\frac{m}{s}$). U slučaju putanje određene na mrežastoj karti zauzeća sa sigurnosnom maskom cijena, takva će putanja biti udaljenija od prepreka pa će kriterijska funkcija biti manje ovisna o koeficijentu α. U tom je slučaju kriterijska funkcija prohodnosti samo dodatno osiguranje za sigurno gibanje robota.



Slika 5.7. *Kriterijska funkcija* $putanja(v,\omega)$.

Iako su simulacijski rezultati zadovoljavajući, ovaj pristup je heuristični i ne jamči konvergenciju prema cilju. Potrebno je odrediti kriterijsku funkciju i naći takvu upravljačku sekvencu da u sljedećem koraku optimizacije postoji barem jedna ostvariva trajektorija koja će smanjiti iznos kriterijske funkcije. Teoretska razrada konvergencije algoritma prema cilju opisana je u poglavlju 6.

Objedinjeni algoritam uspješno uzima u obzir dinamičke promjene u prostoru brzim izračunom nove putanje i izračunom mogućih trajektorija robota s obzirom na dinamička i kinematička ograničenja. Međutim, promjene u prostoru moraju biti spore dinamike da



Slika 5.8. *Globalna kriterijska funkcija* $\Gamma(v, \omega)$.

bi putanja računana u diskretnim intervalima bila važeća, odnosno, da bi se sve prepreke u svakom trenutku smatrale statičnima. U sljedećem odlomku predložena je prilagodba objedinjenog algoritma za izbjegavanje gibajućih prepreka u prostoru mobilnog robota.

5.3 Izbjegavanje sudara s gibajućim preprekama

Veliki je izazov u planiranju gibanja mobilnih robota riješiti probleme izazvane prisutnošću gibajućih prepreka u prostoru. Do sada opisani pristupi izbjegavanja prepreka uspješni su sa statičkim preprekama, ali nisu uvijek učinkoviti u rješavanju problema gibajućih objekata.

Dvije su glavne kategorije pristupa rješavanju toga problema: prostorno–vremenski pristupi i pristupi proširenja potencijalnih polja.

Prostorno–vremenski pristupi uključuju vrijeme kao dodatnu dimenziju modela prostora, u kojem gibajuće prepreke smatraju statičnima [61], [37]. Metoda predstavljena u [21] temelji se na analitičkom izračunu predikcije sudara dvaju gibajućih objekata proizvoljnog oblika. U [52] sudari se računaju pomoću brzo-marširajućeg algoritma propagiranja valne fronte. Metoda brzinske prepreke [35] planira gibanje u prostoru brzina, ali ima problema u učinkovitosti dobivene putanje. u toj se metodi stvara kružni isječak između prepreke i robota, tzv. stožac, koji u prostoru brzina čini brzinsku prepreku. Metoda ima nedostatak što ne uzima u obzir pravi trenutak sudara, nego robot počinje izbjegavati prepreku puno ranije nego što se približi prepreci i time radi preveliki zaobilazak na putanji do cilja. Drugi nedostatak je da smatra da su brzine prepreke konstantne unutar vremenskog intervala robota. Pristup predložen u [137] nastavlja se na metodu brzinske prepreke i rješava neke njene nedostatke. Uzima u obzir da se brzine

gibajućih prepreka kontinuirano mijenjaju, kao što je slučaj u prostorima u kojima se gibaju ljudi. Koristi se izraz dinamičko gibanje prepreka. Prostorno–vremenski pristupi uglavnom pretpostavljaju a priori znanje o brzinama pokretnih prepreka. Međutim, postoje metode koje uključuju i praćenje gibajućih objekata u prostoru [20], [24], [22], [57].

Pristupi proširenja potencijalnih polja generiraju umjetnu silu koja odbija robota od pokretnih prepreka [59], [29], [40]. Feder i Slotine [29] koriste harmoničko uniformno potencijalno polje prema [60] i razmatraju tri tipa pokretnih prepreka: prepreke s translacijskom brzinom, prepreke koje se šire i skupljaju i rotirajuće prepreke. Ova metoda uključuje dinamičke prepreke tako što uvodi pojam brzinskog polja računatog prema poznatim brzinama pokretnih prepreka. Metode [63] i [85] temelje se na sličnom pristupu uključivanja informacije o brzinama pokretnih prepreka u proračun potencijalnog polja. Drugi pristup koji ne zahtijeva poznate brzine prepreka, računa repulzivni potencijal prema relativnoj poziciji i brzini robota u odnosu na prepreke [124], [40].

Većina metoda izbjegavanja gibajućih prepreka ima pretpostavku na oblik prepreka. Primjerice, prepreke su modelirane elipsama [52], kvadratima [20], poligonima [37], itd.

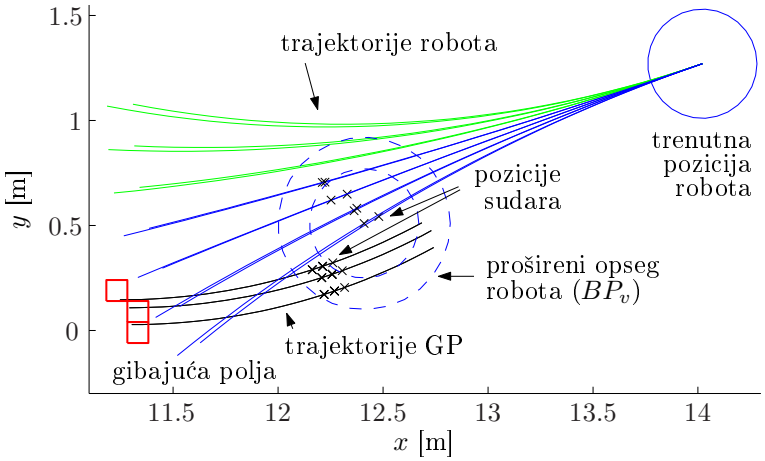
U ovome se radu predlaže algoritam koji pripada prostorno–vremenskom pristupu izbjegavanja gibajućih prepreka [115]. Algoritam se zasniva na proširenju objedinjenja algoritma dinamičkog prozora i algoritma planiranja putanje. Izvorni algoritam objedinjenja (opisan u odlomku 5.2.2) omogućava sigurno gibanje robota u djelomično poznatim prostorima sa statičkim preprekama, ali ne osigurava sigurno gibanje robota u prostorima s gibajućim preprekama. Predložena metoda ne radi nikakve pretpostavke na oblik prepreka i ne razmatra gibajuće prepreke kao objekte već kao gibajuća polja (GP) u mrežastoj karti zauzeća. Metoda zahtijeva poznate translacijsku i rotacijsku brzinu v_{gp}, ω_{gp} i smjer gibanja ϑ_{gp} svakog gibajućeg polja. Procjenjivanje tih triju veličina ovdje nije razmatrano, nego se usredotočujemo na sami algoritam planiranja gibanja među gibajućim preprekama. Uz tri modifikacije uvedene u izvorni algoritam objedinjenja, osigurano je sigurno i glatko gibanje među gibajućim preprekama, uz pretpostavku da najveće brzine gibajućih prepreka nisu veće od najveće brzine robota.

5.3.1 Određivanje pozicije sudara s gibajućim poljima

Robot svojim senzorom otkriva pozicije gibajućih prepreka u okruženju. Prema otkrivenim pozicijama gibajućih prepreka, odgovarajuća polja u mrežastoj karti zauzeća postaju zauzeta. Ta su polja proglašena gibajućim poljima. Za svako gibajuće polje određuje se trajektorija koja počinje iz centra gibajućeg polja, koordinata c_{gp}, s poznatim smjerom gibanja ϑ_{gp} i sastoji se od T točaka koje su određene prema njegovoj konstantnoj brzini v_{gp} i ω_{gp} kako slijedi:

$$\begin{aligned} &\tau_i^{gp} = \{q_i^{gp}(0), q_i^{gp}(1), \ldots, q_i^{gp}(T)\} \\ &\text{uz } q_i^{gp}(k+1) = f(q_i^{gp}(k), u^{gp}),\ u^{gp} = [v_{gp}\ \omega_{gp}]^T,\ k = 0, \ldots, T-1, \end{aligned} \tag{5.26}$$

gdje je $q_i^{gp}(0) = [c_{gp}\ \vartheta_{gp}]^T$. Trajektorije gibajućih polja računaju se u svakom koraku iznova s obzirom na vrijednosti predikcija ϑ_{gp}, v_{gp} i ω_{gp}. Za trajektoriju gibajućeg polja postavljen je isti vremenski horizont T kao i za trajektorije dinamičkog prozora. Ovakav pristup znatno pojednostavnuje izračun vremena sudara t_{sudar}, budući da je nužno samo provjeriti sudare točaka istih indeksa k na trajektoriji dinamičkog prozora i trajektoriji gibajućeg polja, kao što je prikazano na slici 5.9 i tako za sve trajektorije gibajućih polja i po svim trajektorijama dinamičkog prozora. Par točaka koje su u sudaru s najmanjim indeksom k određuje vrijeme sudara t_{sudar}, koje se koristi pri izračunu kriterijske funkcije prohodnosti $prohodnost(v, \omega)$ prema (5.18). Pri tome je točka sudara ona točka u kojoj robot dodiruje prepreku svojom proširenom konturom za veličinu BP_v (opisano u odlomku 5.2.1). Ako gibajuće polje ima brzinu jednaku 0, tada njegova trajektorija ostaje u istoj točki, a izračun vremena sudara podudara se s prethodno opisanim u poglavlju 5.2.1.



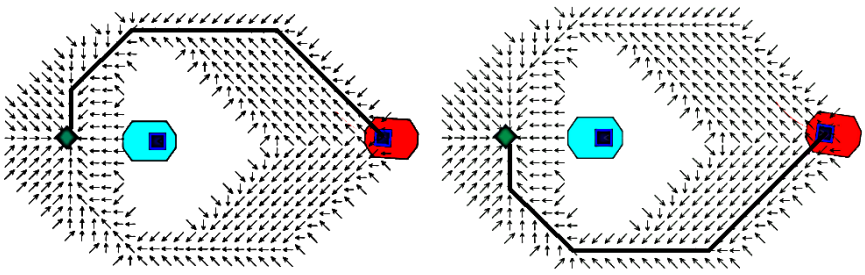
Slika 5.9. *Sudari trajektorija gibajućih polja i dinamičkog prozora.*

5.3.2 Planiranje putanje oko točke sudara

Kada se otkrije sudar k-tih točaka na trajektoriji dinamičkog prozora i gibajućeg polja, gibajuća se polja za algoritam planiranja putanje smatraju slobodnima, a polja u kojima se nalaze točke sudara zauzetima. Algoritam planiranja putanje (algoritam FD*) dinamički planira putanju oko točke sudara, odnosno oko buduće pozicije gibajuće prepreke u karti. Prema tome, pozicija sudara utječe i na kriterijsku funkciju slaganja putanje $putanja(v, \omega)$.

Vrijednosti zauzeća u mrežastoj karti mijenjaju se u svakom koraku u okolici gibajućih prepreka, a prema tome mijenjaju se i smjerovi pokazivača algoritma FD* i određuju nove putanje. Moguća je situacija da FD* u nekoliko vremenskih koraka za redom odredi dvije udaljene putanje vrlo bliskih vrijednosti cijena. Primjerice, u neparnim koracima odredi jedna putanja, a u parnim druga. Na takav odabir utječu učestale

promjene pokazivača u okolini između robota i gibajuće prepreke. Primjer te situacije prikazan je na slici 5.10, gdje se upravljani robot (desno) i gibajuća prepreka (lijevo) gibaju ravno jedan prema drugome. Algoritam FD* postavlja pokazivače između robota

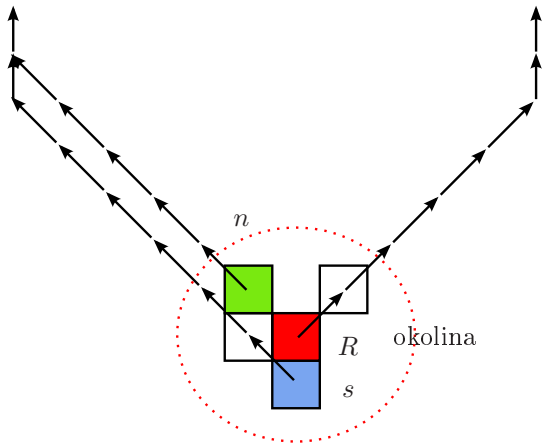


Slika 5.10. *Prebacivanje putanja: lijevo određena putanja u neparnim koracima, desno u parnim.*

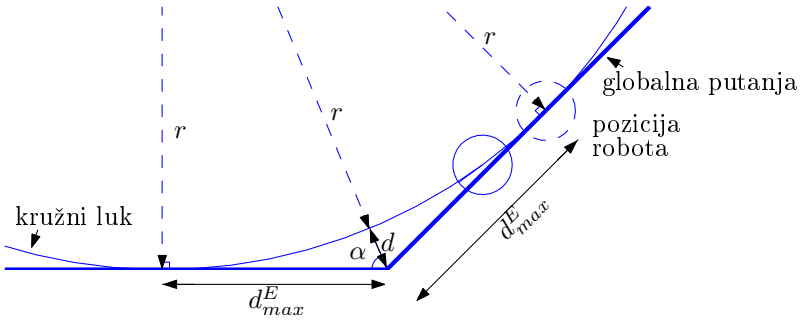
i prepreke koji naizmjence određuju parove simetričnih putanja oko prepreke. Robot će naizmjence pokušati pratiti dvije različite udaljene putanje. Budući da se ne može bočno gibati (zbog neholonomskih ograničenja), nego zajedno s gibanjem prema naprijed zakreće prema putanji, učestalo prebacivanje dovodi do oscilacija u orijentaciji robota, a gibanje prema naprijed dovodi do sudara s gibajućom preprekom.

Opisani problem riješen je razmatranjem putanja iz polja u blizini trenutne pozicije robota, polja R, nazvanih susjednim putanjama. Sve su putanje već određene algoritmom FD*. Među susjednim se putanjama određuje ona koja je slična putanji odabranoj u prošlom koraku (staroj putanji). Sličnost je putanja određena međusobnom udaljenošću polja na putanjama, tako da se uspoređuje k-to polje na razmatranoj susjednoj putanji s k-tim poljem stare putanje. Slična je putanja ona koja ima najmanji zbroj međusobnih udaljenosti polja. Na slici 5.11 prikazan je primjer odabira slične putanje. Naznačene su stara putanja iz polja s, optimalna putanja iz polja R i slična putanja iz polja n. Ako optimalna putanja iz polja R nije bliska staroj putanji, prednost se daje susjednoj putanji sličnoj staroj putanji. Međutim, mora se provjeriti cijena sličnoj putanji, odnosno cijena g^* početnom polju slične putanje n. Ako je cijena $g^*(n)$ veća od cijene trenutnog polja robota $g^*(R)$ uvećana za cijenu prijelaza između tih dvaju polja $g^*(n, R)$, odabire se optimalna putanja iz trenutnog polja robota R. Ovom se provjerom čuva optimalnost odabira susjedne putanje.

Broj susjednih polja u okolini pozicije robota iz kojih se razmatraju susjedne putanje određen je svojstom objedinjavanja algoritama FD* i dinamičkog prozora, što je objašnjeno u nastavku. Pretpostavimo najgori slučaj u kojem je najveći odmak robota od globalne putanje, prikazan na slici 5.12. Putanja je sastavljena od dugačkih ravnih segmenata i referentna je točka na najvećoj udaljenosti od robota $d^E_{max} = S_{max}$, potičući najveću translacijsku brzinu. Trajektorija je robota niz kružnih lukova i ravnih linija i stoga robot odmiče od točke promjene smjera putanje za neku udaljenost d.



Slika 5.11. *Primjer odabira slične putanje.*



Slika 5.12. *Najveći odmak robota od globalne putanje.*

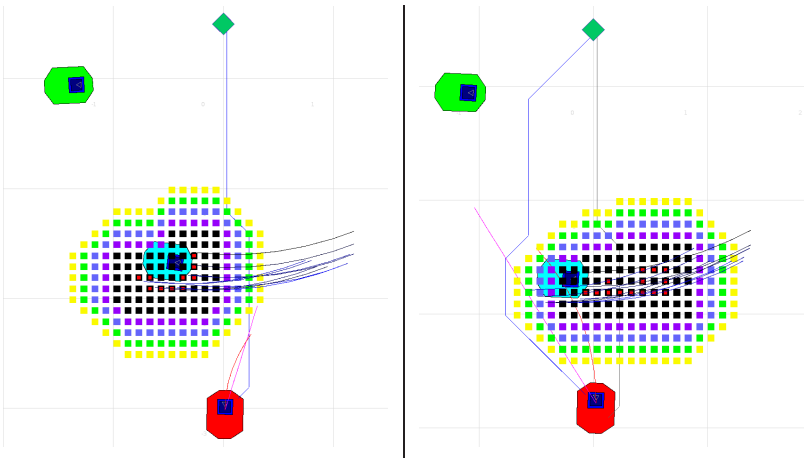
Robot počinje odmicati od globalne putanje u trenutku kad se referentna točka prebaci na sljedeći ravni segment putanje, odnosno kad se robot približi točki promjene smjera putanje tako da je od nje udaljen za manje od d^E_{max}. Najveći odmak od putanje d_{max} dan je s:

$$d_{max} = d^E_{max}\frac{1-\sin\alpha}{\cos\alpha} = r\frac{1-\sin\alpha}{\sin\alpha}, \tag{5.27}$$

gdje je $\alpha = 3 \cdot 45°/2$ (promjene smjera putanje algoritma FD* su višekratnici od 45°), a $r = d^E_{max}\mathrm{tg}\alpha$ polumjer kružnog luka. Rotacijska brzina pri kojoj bi robot putovao po ovom kružnom luku je $\omega_r = \frac{v_{max}}{r}$, ako je ta brzina ostvariva. Zapravo se robot neće gibati po tom kružnom luku zbog diskretnih trenutaka u kojima primjenjuje upravljačke vrijednosti. Robot će početi kasnije zakretati i praviti krivulju koja je bliže globalnoj putanji. Vrijednost d_{max} je zapravo gornja ograda odmaka robota od globalne putanje. Ovaj odmak uzet je u obzir pri gledanju susjednih polja. Sva polja koja su udaljena od robota za manje od d_{max} korištena su kao početna polja za traženje bliske susjedne putanje.

5.3.3 Primjena sigurnosne maske cijena

Zbog odmaka trajektorije robota od globalne putanje moguće je da se robot previše približi gibajućim preprekama i neželjeno uspori ili udari u prepreku. Da bi se izbjegle takve situacije koristi se sigurnosna maska cijena uključena u mrežnu kartu zauzeća, kao što je opisano u poglavlju 2.2.2. Širina sigurnosne maske određena je širinom d_{max} prema (5.27), budući da je to najveći odmak robota od globalne putanje. Broj polja sigurnosne maske jednak je $M_c = \lceil d_{max}/e_{cell} \rceil$. U primjeru prikazanom na slici 5.13, širina sigurnosne maske cijena jednaka je četiri polja. Na slici 5.13, lijevo, prikazan je

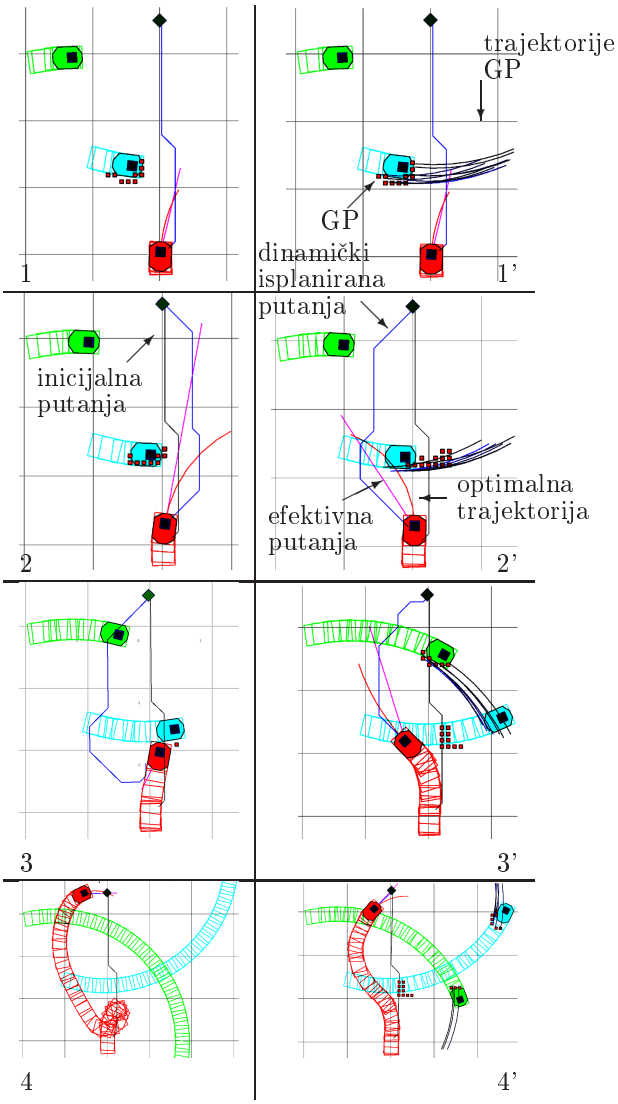


Slika 5.13. *Primjena sigurnosne maske cijena u izbjegavanju gibajućih prepreka.*

trenutak otkrivanja gibajuće prepreke oko koje su bojama naznačeni prijelazi cijena, od najsvijetlije – najmanje vrijednosti do najtamnije – najveće vrijednosti zauzetosti. Središnji crveni kvadratići su gibajuća polja, a oko njih su zauzeta polja (crna boja) unutar maske robota. Algoritam je izračunao buduća mjesta sudara trajektorija gibajućih polja s trajektorijama robota prikazanih na slici 5.13, desno. Ta buduća mjesta sudara upisana su u kartu kao zauzeta polja, oko kojih se računaju zauzeta polja unutar maske robota i vrijednosti sigurnosnih cijena. FD* algoritam izračunat će putanju koja prolazi izvan sigurnosne maske cijena, oko prepreka ako je moguće, ili kroz sredinu uskih prolaza.

5.3.4 Eksperimentalni rezultati

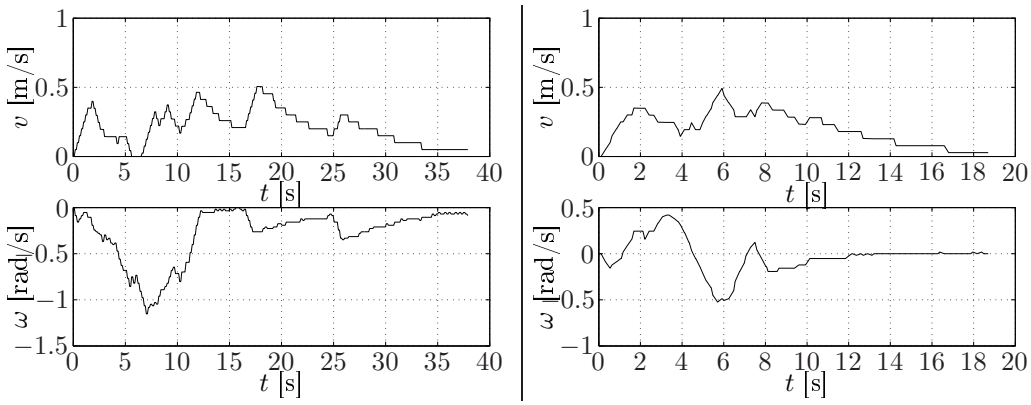
Objedinjeni algoritam testiran je u mnogim situacijama u kojima je robot izveo glatko gibanje bez sudara s gibajućim preprekama. Ovdje su prikazana dva testa algoritma. Rezultati na slici 5.14 prikazuju funkcionalnost algoritma u prvom testu, gdje dva robota predstavljaju gibajuće prepreke koje sijeku putanju upravljanom robotu. Dok lijevi stupac predstavlja gibanje robota ostvareno izvornim objedinjavanjem algoritama FD* i dinamičkog prozora, desni stupac predstavlja gibanje robota ostvareno objedinjenim algoritmom s proširenjima za gibajuće prepreke. Obje metode testirane su pod



Slika 5.14. *Usporedba objedinjenih algoritama: lijevo, izvorni algoritam; desno prošireni algoritam.*

istim uvjetima. Simulacijske scene složene su jedna ispod druge s pripadajućim indeksima. Prva scena predstavlja inicijalnu konfiguraciju u kojoj globalna putanja prolazi ispred prve gibajuće prepreke koja se giba okomito na putanju robota. U drugoj sceni robot upravljan izvornim objedinjenim algoritmom pokušava zaobići prvu gibajuću prepreku najkraćom putanjom s prednje strane gibajuće prepreke, dok prošireni objedinjeni algoritam odabire putanju iza prepreke. U trećoj sceni, robot upravljan izvornim objedinjenim algoritmom zaustavlja se i okreće u mjestu zbog bliže prepreke koja mu brani okretanje u lijevo, dok robot upravljan proširenim objedinjenim algoritmom nastavlja glatkim gibanjem prema cilju. Dinamički isplanirana putanja je kraća s proširenim objedinjenim algoritmom nego s izvornim, zbog uključene predikcije gibanja druge prepreke

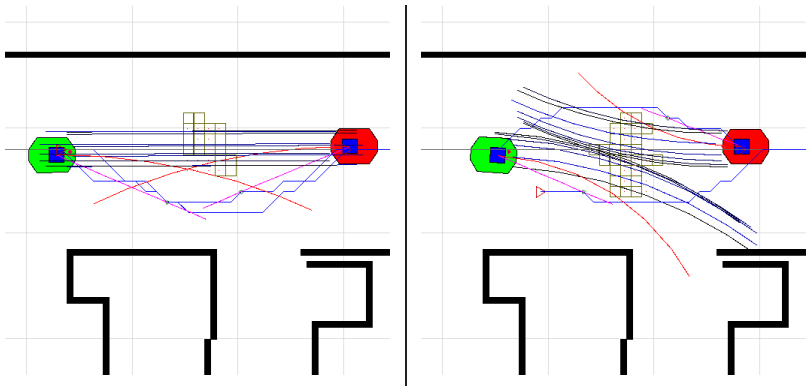
u algoritam planiranja putanje. U prve se tri scene podudaraju vremenski trenutci snimanja rezultata oba algoritma, dok su u četvrtoj sceni prikazane odvožene trajektorije do cilja za oba algoritma. Prikazani kvadratići su mjesta sudara koja su ostala upisana u karti jer se nalaze u nepromatranom dijelu prostora (iza robota). Time se sprječava stalni upis i brisanje iz karte prepreka koje se trenutno ne vide. Prošireni objedinjeni algoritam daje glađu i kraću putanju nego izvorni objedinjeni algoritam. Profili brzina izvornog algoritma prikazani su na slici 5.15, lijevo, a proširenog na slici 5.15, desno. Na slikama je vidljivo da robot dostiže cilj dvaput brže proširenim algoritmom (19 s u



Slika 5.15. *Usporedba algoritama objedinjavanja: Profili brzina izvornog algoritma (lijevo) i proširenog algoritma (desno).*

odnosu na 38 s).

Drugi test algoritma uključuje više robota, od kojih je svaki upravljan istim algoritmom. Roboti međusobno komuniciraju tako što u svakom koraku, nakon što se izvrši upravljački algoritam, objave svoj trenutni položaj x, y, ϑ i trenutnu odluku – upravljačku vrijednost v i ω. Nisu uvedena nikakva "prometna" pravila zaobilaženja niti su zadani prioriteti robotima. Slika 5.16 pokazuje test zaobilaženja dvaju robota u simetričnoj situaciji. Roboti R1 (lijevi) i R2 (desni) kreću se jedan prema drugom.



Slika 5.16. *Zaobilaženje dvaju robota bez prometnih pravila.*

U trenutku k (slika 5.16, lijevo) svaki robot drugoga vidi kao skup gibajućih polja s

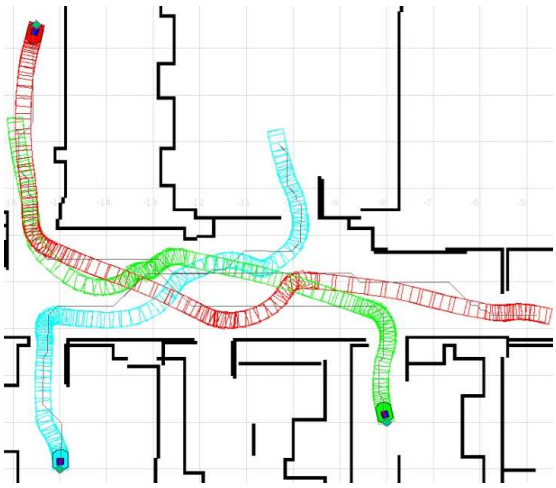
poznatim trajektorijama. R1 i R2 odluče ići istom stranom (donjom), tako što su odredili optimalnu putanju koja zaobilazi mjesto sudara (naznačeni kvadrati na sredini). Već u sljedećem koraku, $k+1$ (slika 5.16, desno) robot R2 promijeni odluku jer mu je putanja s gornje strane jeftinija, budući da su se, zbog prethodne odluke oba robota, mjesta sudara pomakla niže. Međutim u koraku $k+1$ je i robot R1 mogao odrediti kraću putanju s te iste strane i dalje bi se neprestano događalo prebacivanje putanja. Problem oscilacija putanja zapravo je riješen nesinkroniziranim vremenskim trenucima planiranja. Jedan robot uvijek malo ranije otkrije drugog robota i sazna njegovu prošlu odluku. Određene putanje koje su s različitih strana budućeg mjesta sudara dalje se u potpunosti preuzimaju, što je omogućeno procedurom biranja putanje slične staroj putanji iz okoline trenutne pozicije robota. Na slici 5.16, desno, da se primijetiti da putanja robota R1 počinje iz daljeg polja, jer se odabrala putanja slična staroj putanji.

Sljedeći test prikazuje usporedbu izvorne metode objedinjavanja i proširene metode objedinjavanja u interakciji triju robota s istim upravljačkim algoritmom. Robotima su zadani različiti ciljevi, tako da im se isplanirane putanje međusobno sijeku. Napomenimo da nisu uvedena nikakva prometna pravila u gibanjima robota. U oba slučaja svi roboti dostižu svoje ciljeve u konačnom vremenu i bez sudara s drugim robotima i preprekama. Tragovi robota na slikama 5.17 i 5.18 prikazuju glađe odvožene trajektorije robota s proširenim algoritmom u odnosu na izvorni. Roboti upravljani izvornim

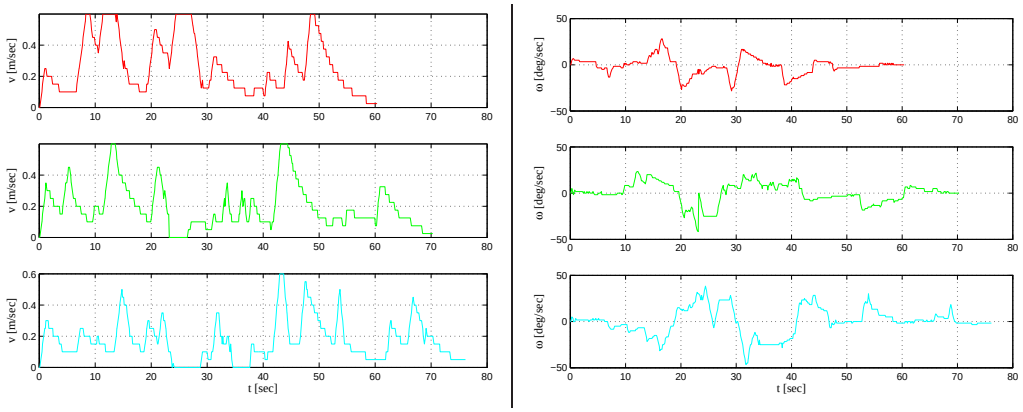
Slika 5.17. *Putanje triju robota upravljanih izvornim objedinjenim algoritmom.*

objedinjenim algoritmom imaju lošije rezultate u brzini postizanja zadanog cilja. Često se zaustavljaju na putanji do cilja i vrte u mjestu, jer druge robote smatraju nepoznatim statičkim preprekama. Profili brzina na slici 5.19 u usporedbi s 5.20 pokazuju očekivanu prednost proširenog algoritma među gibajućim preprekama.

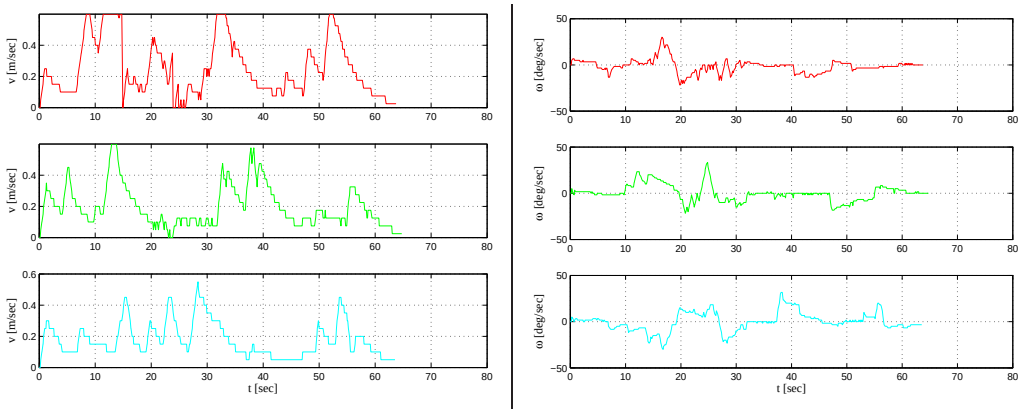
U ovom testu svi roboti uspješno dostižu svoje ciljeve. Međutim, moguće je složiti scenarij u kojem je mjesto budućeg sudara na uskom prolazu tako da sprječava i jednom i drugom robotu nastavak gibanja. Takav problem nazivamo *zastojima u uskim prolazima*



Slika 5.18. *Putanje triju robota upravljanih proširenim objedinjenim algoritmom.*



Slika 5.19. *Profili brzina zaobilaženja triju robota upravljanih izvornim objedinjenim algoritmom.*



Slika 5.20. *Profili brzina zaobilaženja triju robota upravljanih proširenim objedinjenim algoritmom.*

(engl. *deadlock, conflict resolution*). U literaturi se taj problem još naziva i zastojima putanja (engl. *path deadlock*). U sljedećem odlomku predložena je strategija koja rješava problem zastoja u uskim prolazima.

5.4 Izbjegavanje zastoja u uskim prolazima

Zastoj putanje moguća je situacija u planiranju gibanja u kojoj se rješenje ne može naći, iako ono postoji. Tipično je uzrokovana robotima koji jedan drugom blokiraju putanje, a algoritam ne može pronaći rješenje deblokade putanja [90]. U literaturi se ovaj problem javlja u koordinaciji više mobilnih robota u prostoru.

Postoje dva pristupa koordiniranja više robota u prostoru za njihovo postizanje ciljne pozicije: *centralizirani pristup* i *decentralizirani pristup*.

Centralizirani pristup pretpostavlja korištenje centralnog nadgledanja i upravljanja, kojim se planira gibanje svih mobilnih robota uz sprječavanje sukoba s drugim robotima na njihovim putanjama. Brojni centralizirani pristupi koriste shemu upravljanja prometom u mreži. Kao primjer, metoda u [70] koristi takvu shemu kojom se, iz skupa inicijalno izračunatih mogućih putanja odabiru one koje osiguravaju najkraće trajanje gibanja svih mobilnih robota bez sukoba s drugim robotima. Nedostatak centraliziranog pristupa je eksponencijalni porast složenosti problema s brojem upravljanih mobilnih robota.

Nasuprot centraliziranom pristupu, u decentraliziranom se pristupu putanje robotima planiraju zasebno, a nakon toga se primijenjuje određena strategija za rješavanje mogućih sukoba na njihovim putanjama. Ove su metode nekompletne, u smislu da ne jamče pronalazak rješenja iako ono postoji. Popularni pristup je planiranje u prostorno-vremenskom konfiguracijskom prostoru za svakog robota, uz pretpostavku da su u svakom trenutku poznati položaji drugih robota [28]. Metode tog pristupa pridjeljuju prioritete mobilnim robotima. Gibanja robota planiraju se prema redoslijedu zadanom prioritetima, tako što se trajektorija robota nižeg prioriteta računa uzimajući u obzir već izračunate trajektorije svih robota viših prioriteta [15], [87], [73]. Metoda koja se zasniva na ponašanju kolonija mrava [74] rješava zastoje na putanjama naknadno, nakon što su putanje izračunate. Pritom se koristi pretpostavka da se roboti mogu međusobno otkriti senzorima i saznati vrijednosti brzina otkrivenih robota. Veći prioritet pridjeljen je onom robotu koji ima veću brzinu. Nedostatak je spomenutih pristupa što dodjeljivanje prioriteta robotima ne jamči uspješnost metode u svim situacijama [7]. Brojni su pristupi zasnovani na ponašanjima koji primijenjuju teoriju odlučivanja u izbjegavanju prepreka i sprječavanju zastoja [100], [99]. U [71] koriste koordinacijske grafove za izbjegavanje zastoja u uskim prolazima. Metoda pati od velike računske složenosti koja raste s brojem uskih prolaza. Koordinacijski se graf konstruira za svaki uski prolaz. To je tehnika iz područja umjetne inteligencije za odlučivanje i planiranje kooperativnih više-agentnih dinamičkih sustava [45].

Opisane su metode vezane uglavnom za više robotske sustave koji ne uključuju ljude.

U literaturi postoje brojne metode koje su dizajnirane za navigaciju među ljudima i interakciju s ljudima. Primjerice, robot RHINO [17] dizajniran je kao muzejski vodič. Ima čitavu hijerarhijsku strukturu različitih algoritama koji rješavaju pojedine podzadatke. Problem zastoja rješava brzim odabirom drugih lokalnih putanja, a ako su zastoji uzastopni, preračunava novu globalnu putanju. Slično ponašanje ima i robot muzejski vodič MINERVA [129].

U ovom se radu predlaže strategija rješavanja problema zastoja u uskim prolazima prouzrokovang mobilnim robotima, ljudima i drugim gibajućim preprekama. Rješava se problem uz najstroži uvjet: robot s gibajućim preprekama ne može komunicirati već samo opaža njihovu prisutnost svojim senzorom. Problem zastoja rješen je jednostavnom strategijom kao što je slučajna varijacija u sprječavanju zastoja u ethernet mrežama [8]. Eksperimentalni rezultati na stvarnom robotu i sustavni simulacijski testovi osmišljeni su tako da se provjeri uspješnost predložene strategije u rješavanju zastoja u uskim prolazima.

5.4.1 Strategija izbjegavanja zastoja

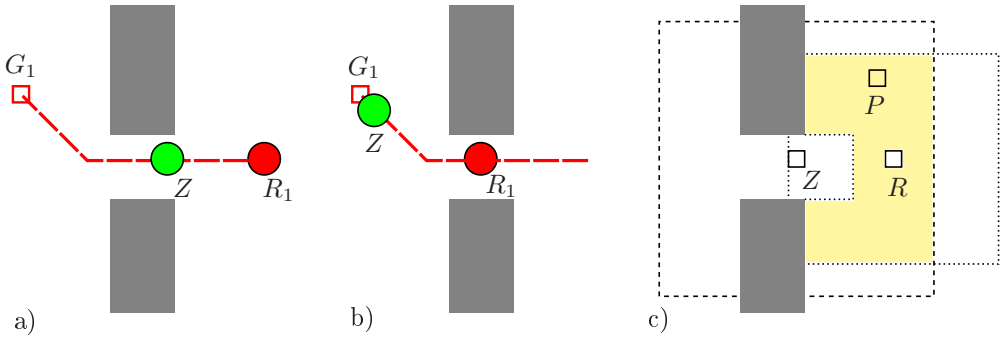
Predložena strategija sastoji se od sljedećih koraka:

Otkrivanje situacije zastoja Uski prolazi izolirani su iz mrežaste karte zauzeća metodom opisanom u odlomku 3.4.1. Robot svojim senzorom otkriva koja su polja u mrežastoj karti zauzeća postala zauzeta. Ako se ta polja nalaze u skupu uskih prolaza $\mathcal{U}$, to znači da se novo otkrivena prepreka u prostoru nalazi u uskom prolazu. Zastoj u uskom prolazu može se dogoditi u dva slučaja:

a) prepreka koja se nalazi u uskom prolazu blokira putanju robota (slika 5.21a).

b) robot se nalazi u uskom prolazu, a prepreka blokira putanju (slika 5.21b).

U slučaju b) robot koji se nalazi u uskom prolazu može prouzročiti situaciju a) nekom drugom robotu.



Slika 5.21. *Slučaj zastoja: a) prepreka Z u uskom prolazu blokira putanju robotu R_1; b) robot R_1 je u uskom prolazu, a prepreka Z mu blokira putanju; c) odabir polja promatranja P za slučaj a).*

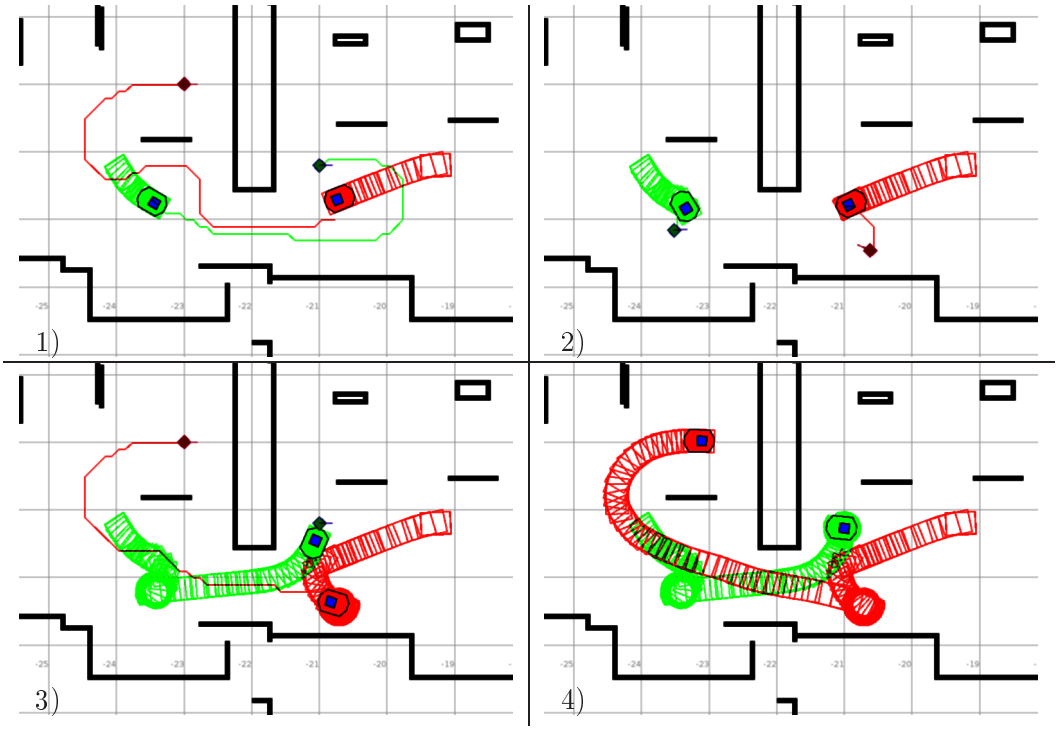
Odabir polja promatranja Da robot svojom pozicijom u zastoju ne bi blokirao putanje drugim gibajućim objektima, odabire se sigurna pozicija u prostoru iz koje robot može promatrati mjesto na kojem je putanja blokirana. Neka je R polje u kojem se robot nalazi, a Z prvo zauzeto polje na blokiranoj putanji, gledajući od polja R. Polje promatranja P odabire se kao bilo koje polje (slučajnim odabirom), koje se nalazi u slobodnom dijelu prostora izvan uskog prolaza, udaljeno od Z manje od dometa senzora S_{max} i udaljeno od R manje od $\|Z - R\|$ (slika 5.21c). Radi jednostavnosti koristi se L1 norma. Zadnjim uvjetom se sprječava odabir polja promatranja s druge strane prolaza. Dodatno, mora biti ispunjena vidljivost između P i Z. Moguće je da, zbog napučenosti prostora drugim gibajućim objektima, polje P postane u međuvremenu zauzeto. Tada se postupak odabira polja P ponavlja.

Čekanje i ponovni pokušaj Robot u polju promatranja P čeka slučajan broj vremenskih intervala $T_z \in \{0, \ldots, T_{zmax}\}$. Za vrijeme čekanja stalno prati zauzetost polja Z. Ako se za vrijeme cjelokupnog čekanja prolaz nije oslobodio, a postoji alternativna putanja do cilja, odustaje od prvotne putanje i odabire novu. Ako ne postoji alternativna putanja i ako je prvotna putanja i dalje blokirana, nakon T_z broja intervala odabire novi slučajan broj iz dvaput šireg skupa mogućih intervala $T_z \in \{0, \ldots, 2T_{zmax}\}$. Dalje se svakim pokušajem dvostruko proširuje skup mogućih intervala. Nakon što istekne maksimalni broj pokušaja, odustaje od putanje i odabire novi cilj. Prolaz se može osloboditi čim robot stigne u polje promatranja. Međutim, čekanje slučajnog broja intervala nužno je za rješavanje situacije u kojoj dva robota, upravljana istim algoritmom, jedan drugom blokiraju putanje. U tom slučaju bi svaki robot otišao u svoje polje promatranja i prolaz bi bio slobodan. Tada bi krenuli u isto vrijeme i zastoj bi se ponovo dogodio.

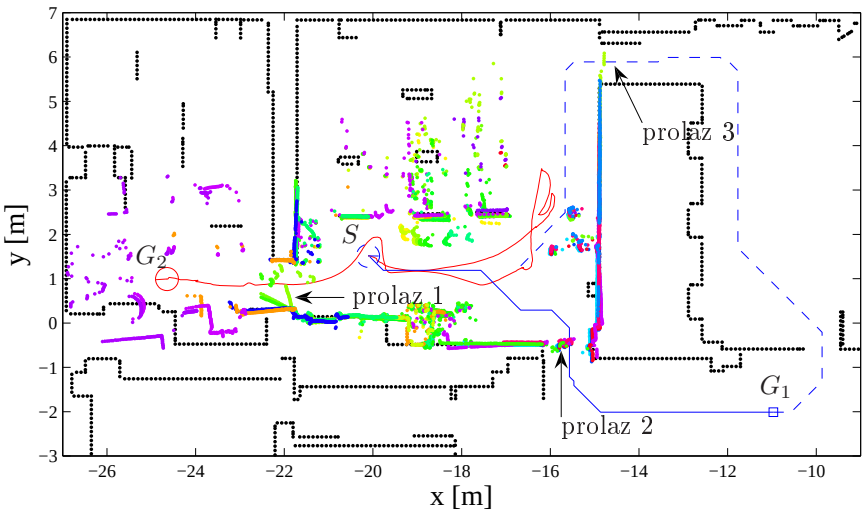
5.4.2 Testiranje strategije izbjegavanja zastoja

Na slici 5.22 prikazana su četiri isječka iz simulacijskog testa izbjegavanja zastoja u uskom prolazu, u kojem su dva robota upravljana istim algoritmom. Putanje koje roboti slijede prikazane su odgovarajućom bojom: zeleni (svijetlosivi) robot slijedi zelenu (svijetlosivu) putanju, a crveni (tamnosivi) robot slijedi crvenu (tamnosivu) putanju. Kad se roboti međusobno otkriju senzorima, putanje se preračunavaju tako da zaobilaze očitana mjesta prepreka (isječak 1). Nedugo zatim, roboti jedan drugom blokiraju putanju i određuju pozicije promatranja s orijentacijama prema blokiranom prolazu (isječak 2), označene s rombovima i crticama koje označavaju orijentacije. Nakon dolaska u svoju poziciju promatranja, svaki robot čeka slučajan broj koraka. Zeleni robot čeka kraće, i svojim nastavkom gibanja do cilja blokira prolaz crvenom robotu (isječak 3). Crveni robot nastavlja gibanje do cilja kad mu se oslobodi prolaz (isječak 4).

Na slici 5.23 prikazan je rezultat eksperimenta na stvarnom robotu među gibajućim preprekama – ljudima koji prolaze kroz prolaze i otvaraju i zatvaraju vrata [26]. Eksperiment je osmišljen tako da inicijalna putanja do cilja G_1 prolazi kroz donji prolaz



Slika 5.22. *Izbjegavanje zastoja dva robota u uskom prolazu.*



Slika 5.23. *Izbjegavanje zastoja na stvarnom prostoru.*

2, na kojem stoje ljudi, a postoji alternativna putanja (označena crtkanom linijom) kroz prolaz 3, koji ima zatvorena vrata. Robot pokušava proći kroz prolaz 2, budući da je trajno zauzet odabire alternativnu putanju, koja prolazi kroz prolaz 3. Budući da je i prolaz 3 trajno zauzet, opet odabire prvu putanju, kroz prolaz 2. Nakon što istekne maksimalni broj pokušaja (pet pokušaja), odustaje od prvotnog cilja G_1 i odabire novi,

G_2. Putanja do cilja G_2 prolazi kroz prolaz 1, koji ima zatvorena vrata. Međutim, vrata se ubrzo otvaraju i nakon čekanja iz pozicije promatranja, robot nastavlja do cilja G_2. Odvoženja putanja prikazana je krivudavom linijom, a laserska očitanja u pojedinim trenucima točkama različitih boja, tako da se mogu zamijetiti tragovi otvaranja vrata u prolazu 1 i gibanja ljudi u prostoru.

5.5 Sažetak

U ovom su poglavlju opisane metode slijeđenja isplanirane putanje zasnovane na algoritmu dinamičkog prozora koje proračunavaju moguće trajektorije s obzirom na kinematička i dinamička ograničenja robota. Predložen je kriterij objedinjavanja algoritma planiranja putanje i algoritma dinamičkog prozora, tako što se uspoređuju trajektorije dinamičkog prozora s globalnom putanjom. Razvijenim objedinjenim algoritmom planiranja gibanja jamči se sigurno gibanje robota do globalnog cilja u prisustvu statičkih prepreka, ali ne i dinamičkih (gibajućih) prepreka.

Uz određene modifikacije uvedene u izvorni objedinjeni algoritam planiranja gibanja, osigurano je sigurno i glatko gibanje među gibajućim preprekama, uz pretpostavku da najveće brzine gibajućih prepreka nisu veće od najveće brzine robota. Gibanje prepreke predstavljeno je gibanjem zauzetih gibajućih polja u mrežastoj karti zauzeća. Predviđene trajektorije svakog gibajućeg polja korištene su u izračunu sudara s trajektorijama dinamičkog prozora. Algoritmom FD* računa se nova putanja oko pozicije budućeg sudara s preprekom. Korištenjem sigurnosne maske cijena, dodatno se odmiče nova putanja od budućeg mjesta sudara. Problem uzastopnog prebacivanja između dviju različitih putanja u simetričnim situacijama, koje su česte u zaobilaženju gibajućih prepreka, riješen je odabirom odgovarajuće putanje iz okoline trenutne pozicije robota. Pritom se čuva optimalnost odabrane putanje.

Eksperimenti potvrđuju prednost proširenog objedinjenog algoritma planiranja gibanja nad izvornim algoritmom u učinkovitom gibanju robota među gibajućim preprekama. Dodatno, ostvareno je učinkovito gibanje više robota koji su upravljani proširenim objedinjenim algoritmom bez uvođenja prometnih pravila i uz pretpostavku da svaki robot zna zadnju odluku drugih robota.

Da bi prošireni objedinjeni algoritam planiranja gibanja bio primijenjiv u dugotrajnom gibanju robota kroz prostor popunjen mnogim gibajućim preprekama uvedena je strategija koja rješava problem zastoja u uskim prolazima. Zastoj se javlja u situacijama kada se robot i neka gibajuća prepreka (drugi robot, čovjek) sretnu u uskom prolazu i jedno drugom onemoguće nastavak gibanja prema cilju. Uz pretpostavku najstrožeg uvjeta, tj. da robot s gibajućim preprekama ne može komunicirati već njihovo ponašanje samo opaža svojim percepcijskim senzorima, predložena strategija, zasnovana na konceptu sprječavanja zastoja u ethernet mrežama, uspješno rješava zastoje u uskim prolazima. Uspješnost strategije provjerena je eksperimentima na stvarnom robotu.

Poglavlje 6

Planiranje gibanja s ograničenjima zasnovano na pomičnom horizontu

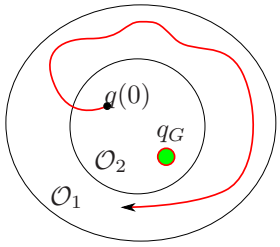
6.1 Pojmovi stabilnosti

Definicija 6.1 (Ravnotežna točka i stabilnost po Ljapunovu). *Neka je sustav dan s*

$$q(i+1) = f(q(i)), \tag{6.1}$$

gdje je $q(i) \in \mathbb{R}^n$, $i \geq 0$, $f : \mathbb{R}^n \to \mathbb{R}^n$ *takva da* $f(\mathcal{Q}) \subset \mathcal{Q}$ *za dani neprazni skup* $\mathcal{Q} \subset \mathbb{R}^n$. *Neka je* $q_G \in \mathcal{Q}$ *ravnotežna točka sustava* (6.1), *tj.* $f(q_G) = q_G$, *i neka je* $\{q(i)\}$, $i \geq 0$, *trajektorija koja zadovoljava* (6.1). *Kažemo da je* q_G *stabilna ravnotežna točka po Ljapunovu ako za bilo koji* $\varepsilon > 0$, *postoji* $\delta > 0$ *takav da*

$$q(0) \in \mathcal{Q} \; i \; \|q(0) - q_G\| < \delta \Longrightarrow \|q(i) - q_G\| < \varepsilon \; za \; svaki \; i \geq 0. \tag{6.2}$$



Slika 6.1. *Stabilnost po Ljapunovu: za bilo koju okolinu* $\mathcal{O}_1$ *postoji okolina* $\mathcal{O}_2$ *takva da trajektorije koje kreću iz* $\mathcal{O}_2$ *ne mogu napustiti* $\mathcal{O}_1$.

Stabilnost po Ljapunovu ne jamči da $q(i)$ konvergira prema q_G kad $i \to \infty$ nego samo da trajektorije sustava ostaju u okolini q_G (slika 6.1).

Definicija 6.2 (Asimptotska stabilnost). *Ravnotežna točka sustava* (6.1), $q_G \in \mathcal{Q}$,

asimptotski je stabilna ako vrijedi:

1. *(stabilnost) q_G je stabilna u $\mathcal{Q}$ po Ljapunovu.*
2. *(konvergencija) postoji $\eta > 0$ takav da*

$$q(0) \in \mathcal{Q} \; i \; \|q(0) - q_G\| < \eta \Longrightarrow \lim_{i\to\infty} q(i) = q_G. \tag{6.3}$$

Definicija 6.2 opisuje lokalnu asimptotsku stabilnost jer je definirana za lokalnu okolinu oko q_G. Globalna inačica asimptotske stabilnosti uključuje cijeli skup $\mathcal{Q}$.

Definicija 6.3. *Ravnotežna točka sustava (6.1), $q_G \in \mathcal{Q}$, globalno je asimptotski stabilna ako je stabilna u $\mathcal{Q}$ po Ljapunovu i ako*

$$q(0) \in \mathcal{Q} \Longrightarrow \lim_{i\to\infty} q(i) = q_G. \tag{6.4}$$

U planiranju gibanja preferira se dostizanje q_G u konačnom vremenu umjesto u beskonačnosti. Ako se cilj opiše malim krugom oko q_G, takav će cilj biti dostignut u konačnom vremenu [69].

6.1.1 Test stabilnosti

Jedna od najčešćih metoda dokazivanja stabilnosti nelinearnih sustava zasniva se na konstrukciji tzv. Ljapunovljeve funkcije [69].

Teorem 6.1 (Test konvergencije po Ljapunovu)**.** *Neka je skup $\mathcal{Q} \subset \mathbb{R}^n$, $f : \mathbb{R}^n \to \mathbb{R}^n$, $f(q_G) = q_G$ i $f(\mathcal{Q}) \subset \mathcal{Q}$. Neka postoji Ljapunovljeva funkcija $V : \mathcal{Q} \to [0, \infty\rangle$ koja zadovoljava sljedeća svojstva:*[1]

(i) $V(\cdot)$ se smanjuje po trajektorijama (6.1) koje počinju u $\mathcal{Q}$ na sljedeći način: postoji kontinuirana funkcija $\gamma : [0, \infty\rangle \to [0, \infty\rangle$, $\gamma(t) > 0$ za sve $t > 0$, tako da

$$V(f(q)) - V(q) \leq -\gamma(\|q - q_G\|) \text{ za svaki } q \in \mathcal{Q}, \tag{6.5}$$

(ii) Za svaku neograničenu trajektoriju $\{y(i)\} \subset \mathcal{Q}$ postoji neki j takav da vrijedi

$$\lim_{i\to\infty} \sup V(y(i)) > V(y(j)). \tag{6.6}$$

Tada ako je $q_G \in \mathcal{Q}$, q_G je globalno konvergentna točka u $\mathcal{Q}$.

Dokaz: Dokaz je dan u [44].

Teorem 6.2 (Test stabilnosti po Ljapunovu)**.** *Neka skup $\mathcal{Q} \subset \mathbb{R}^n$ sadrži okolinu točke q_G, $\mathcal{O}_\eta = \{q \in \mathbb{R}^n \mid \|q - q_G\| < \eta\}$. Neka je $f : \mathbb{R}^n \to \mathbb{R}^n$, $f(q_G) = q_G$ i $f(\mathcal{Q}) \subset \mathcal{Q}$. Neka*

[1] Svojstvo (ii) se može preskočiti ako je $\mathcal{Q}$ ograničen skup.

postoji Ljapunovljeva funkcija $V : \mathcal{Q} \to [0, \infty\rangle$, $V(q_G) = 0$, *koja zadovoljava sljedeća svojstva:*

(i) $V(\cdot)$ *je kontinuirana na* $\mathcal{O}_\eta$;

(ii) ako je trajektorija $\{y(k)\} \subset \mathcal{Q}$ *takva da* $\lim_{k\to\infty} V(k) = 0$ *tada* $\lim_{k\to\infty} y(k) = q_G$;

(iii) $V(f(q)) - V(q) \leq 0$, *za svaki* $q \in \mathcal{O}_\eta$.

Tada je q_G *stabilna ravnotežna točka za* (6.1) *u* $\mathcal{Q}$.

Dokaz: Dokaz je dan u [44].

6.2 Navigacijska funkcija

Neka je q_G ciljno stanje u konfiguracijskom prostoru mobilnog robota $\mathcal{Q}$. Neka je L lokalni operator koji iz nekog stanja q određuje susjedno stanje $q' = L(q)$ koje ima najmanju vrijednost funkcije ϕ od svih susjednih stanja (stanje s najmanjom vrijednosti cijene od svih stanja iz skupa koji vraća funkcija sljednika S, 2.1). Funkcija $\phi : \mathcal{Q} \to \mathbb{R}$ zove se navigacijska funkcija ako vrijede sljedeća svojstva:

1. $\phi(q_G) = 0$
2. $\phi(q) = \infty$ ako i samo ako ciljno stanje q_G nije dohvatljivo iz q (ne postoji putanja od q do q_G).
3. Za svako dohvatljivo stanje q lokalni operator određuje stanje $q' = L(q)$ za koje je $\phi(q') < \phi(q)$.

U literaturi postoje brojne navigacijske funkcije. Jedna od prvih navigacijskih funkcija jest funkcija umjetnih potencijalnih polja [105]. Funkcija je definirana tako da zadovoljava sva tri navedena uvjeta, međutim ograničena je na prostore s preprekama u obliku diskova. Metode potencijalnih polja općenito promatraju robota kao točku u prostoru na koju djeluju sile odbijanja od prepreka i sile privlačenja k cilju. Većina metoda ima problem lokalnog minimuma koji nije cilj. Potencijalna polja spajaju tri koraka navigacije robota – planiranje putanje, planiranje trajektorije i upravljanje robotom u jedan korak, koristeći upravljački zakon preko negativnog gradijenta funkcije potencijalnih polja. Za metodu se lako pokazuje konvergencija prema cilju, odnosno asimptotska stabilnost po Ljapunovu [105].

Mnoge u literaturi korištene navigacijske funkcije zasnovane su na mreži (diskretne navigacijske funkcije) i mogu se shvatiti kao specijalni slučajevi potencijalnih polja. Već spomenuti primjer je navigacijska funkcija NF1 [68]. Bilo koji algoritam pretraživanja grafa može se iskoristiti za izračun diskretne navigacijske funkcije. Primjerice, Dijkstrin algoritam s ciljem kao početnim čvorom pretraživanja, određuje optimalnu diskretnu navigacijsku funkciju. Prednost je tih metoda što nemaju lokalni minimum, ali nedostatak

im je računanje na diskretnom skupu stanja u prostoru. To bi značilo da trajektorija robota mora prolaziti kroz centre polja. Ako bi se kontinuirana funkcija cijene putanje koristila kao navigacijska funkcija, tada bi trajektorija mogla konvergirati pravom optimalnom rješenju.

U prostorima $\mathbb{R}^2$ može se primijeniti kontinuirana inačica Dijkstrinog algoritma za računanje najkraće Euklidske putanje iz bilo koje točke prostora [83, 49] i tako odrediti kontinuiranu navigacijsku funkciju. Nedostatak je takvog postupka velika računska složenost.

U [86] koriste diskretnu navigacijsku funkciju proračunanu na mreži iz koje interpolacijom određuju kontinuiranu navigacijsku funkciju. Nedostatak te navigacijske funkcije je korištenje L_2 metrike (nema dijagonalnih prijelaza) za proračun cijena putanje u mrežnim točkama. Metoda izložena u [95, 96] koristi brzomaršir ajući algoritam [118, 23] kojim određuje diskretnu navigacijsku funkciju u kojoj su vrijednosti cijena Euklidske najkraće putanje do cilja. Iz takve diskretne navigacijske funkcije računa se interpolacijom kontinuirana navigacijska funkcija. Glavna prednost te metode je glatka navigacijska funkcija, što poboljšava kvalitetu putanje robota za vrijeme silaženja po gradijentu. Nedostatak je velika računska složenost interpolacije, što se nadoknađuje svojstvom dinamičkog planiranja (brzog replaniranja u dinamičkim prostorima).

Ljapunovljeve su funkcije usko povezane s navigacijskim funkcijama [69]. U diskretnim navigacijskim funkcijama, svaki prijelaz iz aktivnog polja u polje s najmanjom cijenom putanje do cilja znači smanjenje cijene putanje do cilja (u novom stanju robota) sve dok se ne dosegne cijena 0 u cilju G. I navigacijska funkcija i Ljapunovljeva funkcija osiguravaju da trajektorije ne budu zaustavljene u lokalnom minimumu. Ljapunovljeve su funkcije općenitije od navigacijskih funkcija jer ne zahtijevaju optimalnost. Svojstva navigacijske funkcije vrlo su slična pozitivno definitnom svojstvu Ljapunovljeve funkcije. Ljapunovljeve funkcije označavaju neki oblik udaljenosti do q_G i stvaraju monotono gibanje prema q_G, koje ne mora biti i optimalno.

U ovome se radu predlaže metoda koja koristi diskretnu navigacijsku funkciju određenu algoritmom D* na mrežastoj karti zauzeća, a interpolacija se računa samo u točkama trajektorija robota. Navigacijska funkcija računa se samo za Kartezijeve koordinate u prostoru, za skup $\mathcal{C} \subset \mathbb{R}^2$. Dodatno, ciljno stanje q_G određeno je Kartezijevim koordinatama (x_G, y_G), a orijentacija ϑ_G je proizvoljna. Pretpostavka je da je pozicija cilja u centru polja mrežaste karte zauzeća, odnosno $(x_G, y_G) = c_G$, gdje je $G \in \mathcal{N}$ ciljni čvor.

6.2.1 Interpolacija cijene putanje

Neka je za zadani ciljni čvor G algoritam D* odredio optimalnu cijenu putanje iz svakog čvora $n \in \mathcal{N}$ do G, označenu s $g^*(n)$ (zbog zadanog cilja G koristimo skraćenicu $g^*(n)$ umjesto $g^*(n, G)$). Za čvorove iz kojih ne postoji putanja do cilja je $g^* = \infty$. Vrijednosti g^* određene su algoritmom D* samo za točke centara polja. Vrijednosti cijene putanje u proizvoljnoj točki prostora mogu se pak dobiti interpolacijom. Neka

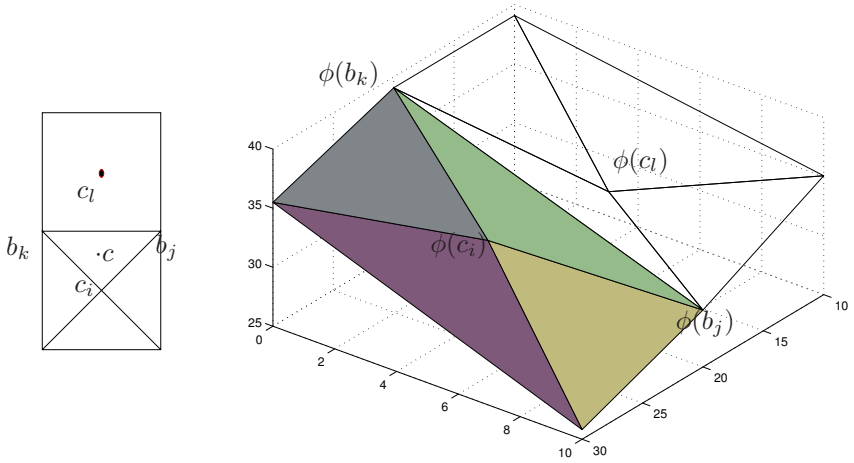
je kontinuirana funkcija cijene putanje $\phi : \mathbb{R}^2 \to \mathbb{R}^2_+$ ostvarena u obliku simplicijalnog kompleksa [109] sastavljenog od standardnih 2-simpleksa nalijepljenih jedan na drugog tako da se nastavljaju bez prekida. Standardni 2-simpleks, označen kao $\triangle^2$, definiran je s

$$\triangle^2 = \left\{\alpha_1 p_1 + \alpha_2 p_2 + \alpha_3 p_3 \in \mathbb{R}^2 \mid \alpha_1, \alpha_2, \alpha_3 \geq 0 \text{ i } \alpha_1 + \alpha_2 + \alpha_3 = 1\right\}, \tag{6.7}$$

gdje su p_1, p_2, p_3 nekolinearne točke u $\mathbb{R}^2$. Svako polje (kvadrat) mrežne karte podijeljeno je na četiri trokuta jednakih površina sa zajedničkim vrhom u centru polja, kao što je prikazano na slici 6.2, lijevo. Da bi se odredila vrijednost ϕ u proizvoljnoj točki $c \in \mathbb{R}^2$, potrebno je odrediti vrijednosti ϕ u najbliže tri točke od c (dva vrha i jedan centar polja, na slici 6.2 označeni s c_i, b_j i b_k). Centri polja imaju svoju g^* vrijednost izračunanu algoritmom D* pa vrijedi $\phi(c_i) = g^*(i)$, $\forall i \in \mathcal{N}$. Skup koordinata vrhova polja definiran je kao $\mathcal{B} = \{b := \frac{c_n + c_m}{2} \mid n, m \in \mathcal{K} \text{ i } \|c_n - c_m\| = d_{polje}\}$. Vrijednost ϕ u vrhovima polja računa se prema vrijednostima cijena u centrima polja koja dodiruju promatrani vrh $b_j \in \mathcal{B}$,

$$\begin{aligned} \phi(b_j) = \tfrac{d_{cell}}{2} + \quad & \min_i \quad g^*(i) \\ & \text{u.u.} \quad i \in \mathcal{N},\ \|b_j - c_i\| = \tfrac{d_{polje}}{2} \end{aligned} \tag{6.8}$$

Vrijednosti u vrhovima mogu se izračunati i algoritmom D*, tako da se svi slobodni



Slika 6.2. *Lijevo: podijela polja na četiri trokuta. Desno: kontinuirana funkcija ϕ kao simplicijalni kompleks.*

vrhovi polja (oni koji dodiruju barem jedno slobodno polje) uključe u skup čvorova u grafu $\mathcal{G}(\mathcal{N}, \mathcal{E}, \mathcal{W})$. Međutim, to nije nužno, budući da su vrijednosti ϕ potrebne samo u točkama trajektorija robota i potrebno je računati ϕ za mali broj vrhova. Vrijednost

ϕ u točki $c \in \mathbb{R}^2$ određuje se rješavanjem sustava triju jednadžbi s tri nepoznanice:

$$\begin{aligned} \alpha_1 c_i + \alpha_2 b_j + \alpha_3 b_k &= c, \\ \alpha_1 + \alpha_2 + \alpha_3 &= 1. \end{aligned} \tag{6.9}$$

Na kraju slijedi

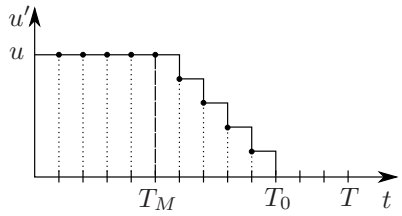
$$\phi(c) = \alpha_1 \phi(c_i) + \alpha_2 \phi(b_j) + \alpha_3 \phi(b_k). \tag{6.10}$$

Primjer kontinuirane funkcije ϕ određene opisanim postupkom za jedno polje prikazana je na slici 6.2, desno.

6.3 Ostvarive trajektorije dinamičkog prozora

Trajektorije korištene u algoritmu dinamičkog prozora kružni su lukovi, određeni iz skupa brzina $\mathcal{V}_d$, kojima je dopušten sudar u nekoj daljoj točki trajektorije. Takve trajektorije nisu ostvarive u cijelom horizontu od T vremenskih intervala, odnosno robot ne smije pratiti takvu trajektoriju duž cijelog horizonta. Ovdje definiramo skup njima ostvarivih trajektorija u trenutku i označen s $\mathcal{X}_i$ i skup odgovarajućih ostvarivih upravljačkih sekvenci $\mathcal{Y}_i$.

Ostvariva je trajektorija kružni luk (ili ravni segment), određen upravljačkom vrijednošću iz diskretiziranog skupa brzina dinamičkog prozora (5.2.1) $u \in \mathcal{V}_d$ i parametrom T_0, koji ne siječe prepreku i čiji je profil brzina određen kako slijedi (slika 6.3. Prvih



Slika 6.3. *Primjer upravljačke sekvence za odabrane $u \in \mathcal{V}_d$ i T_0.*

$T_M + 1$ upravljačkih vrijednosti konstantnog su iznosa $u = [v\ \omega]^T \in \mathcal{V}_d$, a sljedećih T_N linearno su padajuće upravljačke vrijednosti koje čine točke na istom kružnom luku polumjera $r = \frac{v}{\omega}$. Uračunavajući ograničenja (5.9) i (5.10) T_N i T_M određene su kako slijedi:

$$\begin{aligned} T_N &= \max\left\{ \left\lceil \frac{v}{\Delta v_{max}} \right\rceil, \left\lceil \frac{|\omega|}{\Delta \omega_{max}} \right\rceil \right\}, \\ T_M &= T_0 - T_N, \end{aligned} \tag{6.11}$$

gdje je $T_0 \leq T$ vremenski trenutak u kojem se primijenjuje završna upravljačka vrijednost $u_z = [0\ 0]^T$. Da bi trajektorija bila definirana mora vrijediti uvjet $T_N \leq T_0$ tako da vrijedi $T_M \geq 0$. Trajektorija robota τ_i^{u,T_0} i upravljačka sekvenca υ_i^{u,T_0}, za $u \in \mathcal{V}_d$ i

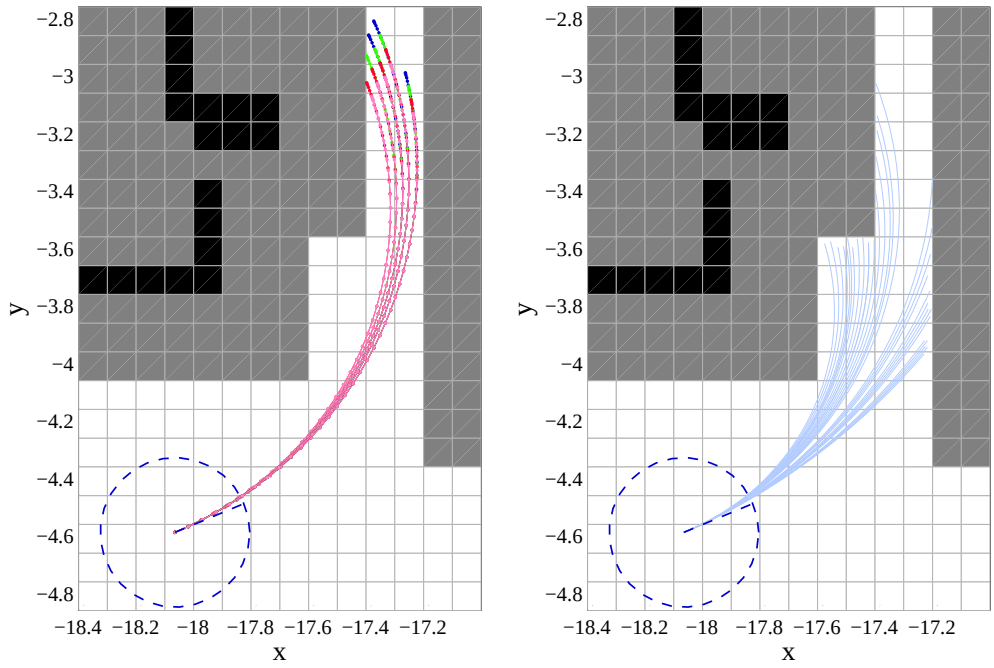
parametar T_0, definirane su u odnosu na trenutni položaj robota $q(i)$ kako slijedi:

$$\begin{aligned} \upsilon_i^{u,T_0} &= \{u_i(0), u_i(1), \ldots, u_i(T-1)\}, \\ \tau_i^{u,T_0} &= \{q_i(0), q_i(1), \ldots, q_i(T)\}, \\ \text{u.u.} \qquad u_i(k) &= \begin{cases} u & \text{za } k = 0, \ldots, T_M, \\ u\dfrac{T_0 - k}{T_N} & \text{za } k = T_M + 1, \ldots, T_0 - 1, \\ u_z & \text{ako } T_0 < T, \text{ za } k = T_0, \ldots, T-1, \end{cases} \\ & q_i(k+1) = f(q_i(k), u_i(k)), \ k = 0, \ldots, T-1, \end{aligned} \tag{6.12}$$

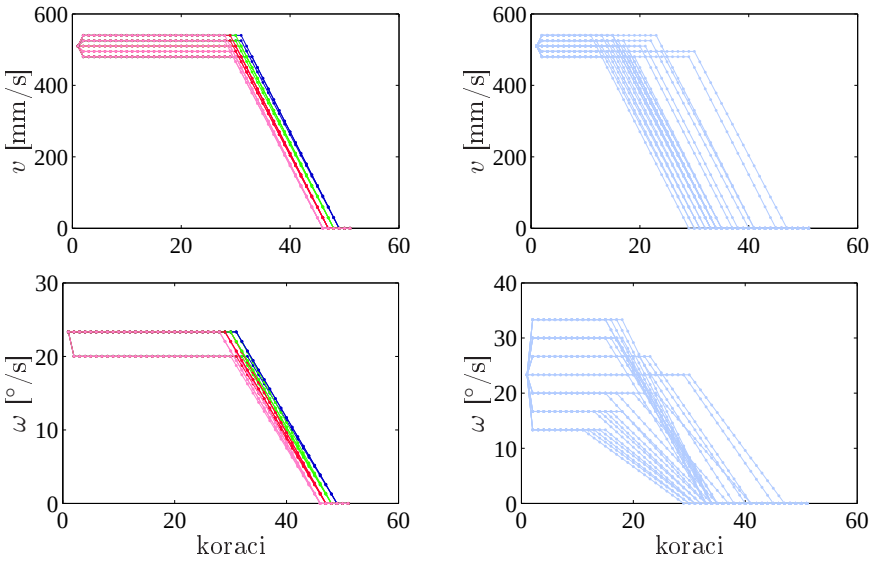
gdje je $q_i(0) = q(i)$ početni položaj robota, a indeks i vremenski trenutak proračuna trajektorije u sljedećih T vremenskih intervala. Budući da je horizont T određen prema dometu senzora S_{max} i najvećoj brzini robota v_{max}, $T = \lfloor \dfrac{S_{max}}{v_{max}\Delta t} \rfloor$, sve će se trajektorije nalaziti unutar okoline o kojoj robot dobiva sve informacije o položaju prepreka u i-tom trenutku planiranja trajektorija. Trajektorije su određene tako da se robot može zaustaviti do kraja horizonta T.

Parametrom T_0 utječe se na duljinu trajektorije unutar horizonta T, odnosno na duljinu niza upravljačkih vrijednosti različitih od nule. Parametar T_0 određuje se prema parametru T_0^* optimalne trajektorije iz prošlog koraka $i-1$. Svaka se trajektorija računa za vrijednosti parametra $T_0 \in \{T_0^* - 2, T_0^* - 1, T_0^*, T_0^* + 1\}$ za koje su zadovoljeni uvjeti $T_0 \leq T$ i $T_N \leq T_0$. Poželjno bi bilo računati trajektorije za sve parametre $T_N \leq T_0 \leq T$, što je računski prezahtjevno. Ograničili smo izbor T_0 na četiri vrijednosti podrazumjevajući da se optimalna trajektorija nalazi u blizini optimalne trajektorije iz prethodnog vremenskog trenutka. Optimalna trajektorija iz koraka $i-1$ pojavljuje se u koraku i kao ostvariva trajektorija kojoj je početna upravljačka vrijednost $u_{i-1}(0) = u_i^*(1)$, a vrijednost $T_0 = T_0^* - 1$. Okolinu te trajektorije prema parametru T_0 čine vrijednosti $T_0 = T_0^* - 2$ i T_0^*. Dodatno još uzimamo i vrijednost $T_0 = T_0^* + 1$ tako da se duljina niza upravljačkih vrijednosti različitih od nule može proširiti na puni horizont T. U slobodnom će se prostoru pri ravnom gibanju parametar T_0, odabirom $T_0 = T_0^* + 1$, postupno povećavati do najvećeg iznosa T, a u zaokretima među preprekama će se smanjivati odabirom $T_0 = T_0^* - 1$ i $T_0 = T_0^* - 2$.

Potrebno je za svaku trajektoriju provjeriti postoji li sudar s preprekom. Provjera sudara s preprekom opisana je u odlomku 6.3.1. Ako sudar za neku trajektoriju postoji, ona je zabranjena. Međutim, moguće je provesti postupak kraćenja trajektorija i odrediti nove ostvarive trajektorije. Trajektorija se ponovno izračunava prema (6.12) za manji broj vremenskih intervala T_0, tako da se postigne kraća kružna trajektorija koja ne sječe prepreku. Pritom mora biti zadovoljen uvjet $T_N \leq T_0$, tako da se robot stigne zaustaviti prije sudara s preprekom. Postupak kraćenja trajektorija daje veći izbor ostvarivih trajektorija u skupu $\mathcal{X}_i$, odnosno upravljačkih sekvenci u skupu $\mathcal{Y}_i$ koje imaju veću vrijednost $u_i(0)$. Bez postupka kraćenja trajektorija odabrale bi se trajektorije koje su kraće zbog manje vrijednosti $u_i(0)$, što bi rezultiralo sporijim gibanjem robota.

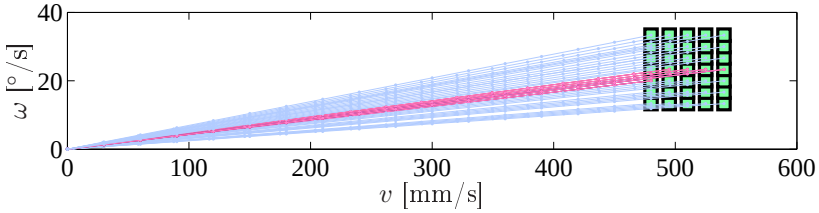


Slika 6.4. *Lijevo: Ostvarive trajektorije prema* $u \in \mathcal{V}_d$ *i četiri vrijednosti parametra* $T_0 \in \{T_0^* - 2, T_0^* - 1, T_0^*, T_0^* + 1\}$*, označene roza* τ_i^{u,T_0^*-2}*, crvenom* τ_i^{u,T_0^*-1}*, zelenom* τ_i^{u,T_0^*} *i plavom bojom* τ_i^{u,T_0^*+1}*; Desno: Skraćene trajektorije koje ne sjeku prepreku.*



Slika 6.5. *Lijevo: Profili brzina ostvarivih trajektorija bez kraćenja; Desno: Profili brzina skraćenih trajektorija.*

Na slici 6.4, lijevo, prikazan je primjer skupa ostvarivih trajektorija bez kraćenja zbog prepreka. Trajektorije su određene prema brzinama iz skupa $\mathcal{V}_d$ i prema četirima vrijednostima parametra $T_0 \in \{T_0^* - 2, T_0^* - 1, T_0^*, T_0^* + 1\}$. Parametar T_0 utječe samo na duljinu trajektorije, tako da se one preklapaju. Pripadajuće su upravljačke sekvence na slici 6.5, lijevo, gdje se mogu razlučiti vrijednosti parametra T_0. Prva po redu prikazana vrijednost brzine je trenutna brzina robota oko koje se računa skup $\mathcal{V}_d$. Može se primijetiti da sve trajektorije imaju istu vrijednost ω. Za ostale vrijednosti $\omega \in \mathcal{V}_d$ trajektorije sjeku prepreku. Na slici 6.4, desno, prikazan je skup skraćenih trajektorija, kojima je parametar T_0 znano smanjen da bi se zaustavile prije prepreke. Pripadajuće su upravljačke sekvence na slici 6.5, desno. Skraćene trajektorije iskorištavaju cijeli skup $\mathcal{V}_d$ i zajedno sa ostalim ostvarivim trajektorijama daju dobru pokrivenost prostora. Na slici 6.6 prikazane su sve ostvarive trajektorije u prostoru brzina. Prikazane su



Slika 6.6. *Prikaz trajektorija u prostoru brzina. Skup $\mathcal{V}_d$ označen je kvadratićima. Žuti pravci odgovaraju skraćenim trajektorijama.*

trajektorije u prostoru brzina pravci koji prolaze kroz ishodište. Pravac označuje da se radi o konstantnom polumjeru kružne trajektorije, odnosno kružnog luka.

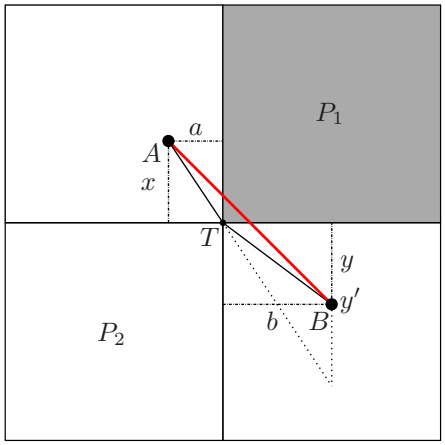
Postupak kraćenja trajektorija može se raditi i kod ulaska u ciljnu poziciju. Prvi je korak odrediti je li cilj dohvatljiv nekom od ostvarivih trajektorija. To se određuje usporedbom duljine trajektorije i vrijednosti ϕ u trenutnoj poziciji robota $R(x(i), y(i))$. Ako je duljina trajektorije veća od ili jednaka $\phi(R)$, tada se postavlja zastavica da je cilj dohvatljiv i od tog se trenutka pa do kraja gibanja svaka trajektorija skraćuje tako da joj duljina bude manja od $\|c_G - R\|$. Time se postiže da niti jedna trajektorija ne prelazi cilj.

6.3.1 Provjera sudara pomoću mrežaste karte zauzeća

Za svaku je točku trajektorije $(x_i(k), y_i(k))$ potrebno provjeriti nalazi li se unutar zauzetog polja[2]. Dodatno, potrebno je provjeriti sjeku li prepreku lukovi između dviju slijednih slobodnih točaka na trajektoriji. Označimo s $a_{A,B}$ luk definiran dvijema slijednim točkama $A(x_i(k), y_i(k))$ i $B(x_i(k+1), y_i(k+1))$ na trajektoriji. Neka je najveća duljina luka $a_{A,B}$ (za najveću translacijsku brzinu) manja od e_{polje} i neka se luk može aproksimirati ravnom linijom (kao što je slučaj u našoj implementaciji). Postoje tri

[2]Podsjetimo se da su stvarne prepreke u mrežastoj karti zauzeća proširene za dimenzije robota tako da su vrhovi prošireni $\sqrt{2}$ puta više nego što je nužno (slika 2.1)

slučaja položaja točaka A i B: 1) obje su u istom polju koje je slobodno; 2) nalaze se u dvama različitim poljima koja se dodiruju stranicom; i 3) nalaze se u dvama dijagonalnim poljima. U prva je dva slučaja jasno da je luk $a_{A,B}$ slobodan, a jedino za treći slučaj treba provjeriti sječe li luk prepreku. Luk može sjeći jedno od dva preostala polja (slika 6.7). Potrebno je provjeriti zauzetost onom polju koje se nalazi s unutarnje strane



Slika 6.7. *Provjera sudara s preprekom.*

tupog kuta koji zatvaraju dužine $\overline{AT}$ i $\overline{TB}$, gdje je točka T zajednički vrh dijagonalnim poljima. Prema slici 6.7 trebaju se provjeriti sljedeće nejednadžbe:

$$\begin{aligned} &y' = \frac{b}{a}x \\ &\text{ako } y < y', \quad \text{provjeriti } P_1 \\ &\text{ako } y > y', \quad \text{provjeriti } P_2 \end{aligned} \tag{6.13}$$

U implementacijama u kojima je najveća duljina luka $a_{A,B}$ veća od e_{polje}, treba napraviti finiju diskretizaciju luka tako da nove točke budu međusobno razmaknute manje od e_{polje}. Tada se na novim točkama primjenjuje opisana metoda provjere sudara.

6.4 Optimizacija pomičnim horizontom uračunavajući ograničenja

Ideja je upravljanja pomičnim horizontom [44] optimirati neku kriterijsku funkciju za fiksni horizont duljine T uračunavajući ograničenja sustava i koristeći trenutno stanje sustava kao početno stanje. Zatim je od dobivene optimalne sekvence od T upravljačkih vrijednosti primjenjuje samo prva vrijednost. Vrijeme zatim ide naprijed jedan korak i isti se optimizacijski problem promatra u fiksnom horizontu duljine T, koristeći novo mjereno stanje sustava kao početno. Tako se kontinuirano mijenja trenutna upravljačka vrijednost na osnovi trenutnog stanja sustava i ograničenja sustava unutar optimizacijskog horizonta duljine T.

Pomični horizont primijenjuje samo prvu optimalnu upravljačku vrijednost i u svakom koraku ponavlja cijelu optimizaciju za sljedećih T koraka horizonta. Svaka optimizacija ovisi o trenutnom mjerenom stanju robota u prostoru. Bez ikakvog poremećaja na sustav (promjene u prostoru mobilnog robota) mjereno stanje bit će jednako onom stanju koje je predviđeno modelom. Ako su model i kriterijska funkcija vremenski nepromijenjivi, tada je jasno da će se ista upravljačka vrijednost u_i primijeniti kada stanje sustava zauzme istu vrijednost. Optimizacija pomičnim horizontom daje vremenski nepromjenjiv upravljački zakon u povratnoj vezi.

U implementaciji pomičnog horizonta za upravljanje gibanjem mobilnog robota definiramo sljedeći optimizacijski problem. U vremenskom trenutku i i za trenutni položaj robota $q(i) = [x(i)\ y(i)\ \vartheta(i)]^T$ rješava se:

$$J^*(q(i)) = \min_{\upsilon_i} \sum_{k=0}^{T} \phi(x_i(k), y_i(k)) + \sum_{k=0}^{T-1} \rho v_i(k), \tag{6.14}$$

uz uvjete:

$$\upsilon_i \in \mathcal{Y}_i, \tag{6.15}$$

$$u_i(k) = [v_i(k)\ \omega_i(k)]^T \in \upsilon_i, \text{ za } k = 0, \ldots, T-1, \tag{6.16}$$

$$q_i(k+1) = f(q_i(k), u_i(k)), \text{ za } k = 0, \ldots, T-1, \tag{6.17}$$

$$q_i(0) = q(i) \tag{6.18}$$

$$q_i(k) = [x_i(k)\ y_i(k)\ \vartheta_i(k)]^T, \text{ za } k = 0, \ldots, T, \tag{6.19}$$

$$\phi(x_i(T), y_i(T)) \leq \phi(x_i(k), y_i(k)), \text{ za } k = 0, \ldots, T-1, \tag{6.20}$$

gdje je ρ mala pozitivna realna vrijednost, $\mathcal{Y}_i$ skup svih ostvarivih upravljačkih sekvenca za trenutno stanje robota i ϕ navigacijska funkcija.

Procedura pomičnog horizonta implicitno definira vremenski nepromjenjiv upravljački zakon $\kappa : \mathcal{Q} \to \mathcal{V}$ u obliku

$$\kappa(q(i)) = u_i^*(0), \tag{6.21}$$

gdje stroga definicija funkcije $\kappa(\cdot)$ zahtjeva da optimalna vrijednost $u_i^*(0)$ bude jedinstvena.

6.4.1 Stabilnost upravljanja pomičnim horizontom

Nije trivijalno pokazati da upravljanje pomičnim horizontom čini zatvoreni sustav stabilnim. Optimizacijski problem koji rješavamo definiran je na konačnom horizontu, dok je stabilnost svojstvo koje mora vrijediti na beskonačnom horizontu. Taj se problem zaobilazi na sljedeći način (a) dodavanjem posebnog otežavanja stanja na kraju horizonta tako da se može uračunati utjecaj događaja iza kraja fiksnog horizonta; (b) uvođenjem terminalnog upravljačkog zakona koji dovodi sustav u ograničeno terminalno područje

unutar optimizacijskog prozora i (c) osiguranjem da je terminalno područje invarijantno u odnosu na terminalni upravljački zakon, što znači da jednom kada stanje dođe u terminalni skup ostaje unutar skupa ako se primijeni terminalni upravljački zakon. Ova su tri uvjeta nužna za uspostavu stabilnosti i nazivaju se terminalnom trojkom [44].

Rezultati stabilnosti implementacije pomičnog horizonta

Funkciju $J^*(q(i))$ optimizacijskog problema (6.14)–(6.20) koristimo kao Ljapunovljevu funkciju da bismo dokazali asimptotsku stabilnost implementacije pomičnog horizonta, gdje funkcija $J^*(q(i))$ ovisi samo o početnom položaju pomičnog horizonta, $q_i(0) = q(i)$. Nadalje, budući da je $\phi(x, y) > 0$ za sve (x, y) osim za cilj (x_G, y_G), za koji je $\phi(x_G, y_G) = 0$, a translacijske brzine nisu nikad negativne, slijedi da je $J^*(q) > 0$ za sve $q \in \mathcal{Q}$ osim za cilj q_G. Iako je sustav $q(i+1) = f(q(i), u(i))$ funkcija položaja $q(i) \in \mathcal{Q}$, planirane su samo pozicije, a orijentacija $\vartheta(i+1)$ je rezultat upravljačke vrijednosti $\omega(i)$. Zbog neholonomskih ograničenja robota orijentaciju određuju pozicija $x(i)$ i $y(i)$ prema (5.6). Prema tome, dolaskom u ciljnu poziciju $x(i) = x_G$, $y(i) = y_G$, vrijednost $\vartheta(i)$ bit će rezultat samog gibanja do cilja. Ako bi se zahtijevala ciljna orijentacija ϑ_G, može se primijeniti zakretanje robota u ciljnoj poziciji, dok robot ne ostvari ciljnu orijentaciju. U nastavku ćemo fiksirati ciljnu orijentaciju ϑ_G i uz primjenu odgovarajuće procedure pokazati asimptotsku stabilnost ciljnog položaja q_G.

Prema [44] potrebne su sljedeće definicije za dokaz stabilnosti.

Definicija 6.4. *Skup $\mathcal{Q}_T$ ostvarivih početnih položaja skup je početnih položaja $q \in \mathcal{Q}$ za koje postoje ostvarive trajektorije i upravljačke sekvence za optimizacijski problem (6.14)–(6.20).*

U ovoj je implementaciji to skup svih položaja iz kojih postoji putanja do cilja, odnosno $\mathcal{Q}_T = \{q = [x\ y\ \vartheta]^T \mid q \in \mathcal{Q} \text{ i } \phi(x, y) < \infty\}$.

Definicija 6.5. *Skup $\mathcal{Q}$ pozitivno je invarijantan za sustav $q(i+1) = f(q(i), u(i))$ uz primijenjenu upravljačku vrijednost $u(i) = \kappa(q(i))$, ako je $f(q, \kappa(q)) \in \mathcal{Q}$ za svaki $q \in \mathcal{Q}$.*

U ovoj je implementaciji skup slobodnih pozicija $\mathcal{C} \subset \mathbb{R}^2$ ograničen rubovima unutarnjeg prostora mobilnog robota, a skup kutova $\Theta \subset \mathbb{R}$ ograničen je vrijednostima $0° - 360°$. Prema tome skup $\mathcal{Q} = \mathcal{C} \times \Theta$ također je ograničen. Dalje, određivanjem ostvarivih trajektorija i ostvarivih upravljačkih sekvenci prema (6.12), koje ne dopuštaju da se položaj robota q nađe u prepreci, ostvarili smo pozitivnu invarijantnost cijelog skupa $\mathcal{Q}$ za f. Za ograničen i pozitivno invarijantan skup $\mathcal{Q}$ za f, prema definiciji stabilnosti po Ljapunovu 6.1 slijedi da je $q_G \in \mathcal{Q}$ stabilna ravnotežna točka za f.

Teorem 6.3. *Za sustav*

$$q(i+1) = f(q(i), u(i)), \ \text{za } i \geq 0, \ f(q_G, u_z) = q_G, \tag{6.22}$$

upravljan algoritmom pomičnog horizonta (6.14)–(6.20) vrijedi:

(i) skup $\mathcal{Q}_T$ ostvarivih početnih položaja pozitivno je invarijantan za zatvoreni sustav.

(ii) q_G je globalno asimptotski stabilan u $\mathcal{Q}_T$ za zatvoreni sustav.

Dokaz: (i) Pozitivna invarijantnost skupa $\mathcal{Q}_T$

Neka je $q(i) \in \mathcal{Q}_T$. Pretpostavimo da u trenutku i i za trenutno stanje $q(i)$, algoritam pomičnog horizonta rješava optimizacijski problem $J^*(q(i))$ prema (6.14)–(6.20) i dobiva optimalnu upravljačku sekvencu i trajektoriju:

$$\upsilon_i^* = \{u_i^*(0), u_i^*(1), \ldots, u_i^*(T-1)\}, \tag{6.23}$$

$$\tau_i^* = \{q_i^*(0), q_i^*(1), \ldots, q_i^*(T)\}. \tag{6.24}$$

Tada se na sustav (6.22) primijeni prvi element upravljačke sekvence (6.23),

$$u(i) = \kappa(q(i)) = u_i^*(0). \tag{6.25}$$

Neka je $q(i+1) = f(q(i), \kappa(q(i)))$ sljedeći položaj robota u trenutku $i+1$. Prema (6.12) postoji ostvariva upravljačka sekvenca $\upsilon_{i+1}^{u,T_0} \in \mathcal{Y}_{i+1}$ i pripadna trajektorija $\tau_{i+1}^{u,T_0} \in \mathcal{X}_{i+1}$, za $T_0 = T_0^* - 1$ i $u = u_i^*(1) \in \mathcal{V}_d$ koje nisu nužno optimalne u izračunu pomičnog horizonta $J^*(q(i+1))$ i za koje vrijedi:

$$\upsilon_{i+1} = \{u_{i+1}(0), u_{i+1}(1), \ldots, u_{i+1}(T-1)\}, \tag{6.26}$$

$$\tau_{i+1} = \{q_{i+1}(0), q_{i+1}(1), \ldots, q_{i+1}(T)\}, \tag{6.27}$$

$$\begin{aligned} &u_{i+1}(k) = u_i^*(k+1), \text{ za } k = 0, \ldots, T-2, \\ &u_{i+1}(T-1) = [0\ 0]^T, \\ &q_{i+1}(k) = q_i^*(k+1), \text{ za } k = 0, \ldots, T-1, \\ &q_{i+1}(T) = q_i^*(T). \end{aligned} \tag{6.28}$$

Prema tome $q(i+1) \in \mathcal{Q}_T$, što pokazuje da je $\mathcal{Q}_T$ pozitivno invarijantan skup za sustav s povratnom vezom $q(i+1) = f(q(i), \kappa(q(i)))$.

(ii) Globalna asimptotska stabilnost

Budući da je $\phi(x_G, y_G) = 0$ tada optimalna upravljačka sekvenca u (6.14)–(6.20) za $q = q_G$ ima sve elemente jednake 0 pa je stoga i $J^*(q_G) = 0$. Stoga $\kappa(q_G) = u_z$. Budući da je $f(q_G, u_z) = q_G$ tada je q_G ravnotežna točka za zatvoreni sustav $q^+ = f(q, \kappa(q))$.

Budući da je $q_G \in \mathcal{Q}$ stabilna ravnotežna točka potrebno je još pokazati za J^* da vrijedi konvergencija prema (6.5) teorema 6.1, čime će se dokazati globalna asimptotska stabilnost zatvorenog sustava.

Za optimalnu upravljačku sekvencu i trajektoriju u i-tom koraku (6.23) i (6.24)

vrijednost Ljapunovljeve funkcije je

$$J^*(q(i)) = \sum_{k=0}^{T} \phi(x_i^*(k), y_i^*(k)) + \sum_{k=0}^{T-1} \rho v_i^*(k). \tag{6.29}$$

Neka su u koraku $i+1$ za sljedeći položaj $q(i+1) = f(q(i), \kappa(q(i)))$ ostvarive i ne nužno optimalne upravljačka sekvenca i trajektorija dane izrazima (6.26)–(6.28). Prema optimalnosti znamo da je

$$J^*(q(i+1)) \leq \sum_{k=0}^{T} \phi(x_{i+1}(k), y_{i+1}(k)) + \sum_{k=0}^{T-1} \rho v_{i+1}(k). \tag{6.30}$$

Dalje, kombinirajući (6.29), (6.28) i (6.30) proizlazi:

$$\begin{aligned} J^*(q(i+1)) &\leq \sum_{k=1}^{T} \phi(x_i^*(k), y_i^*(k)) + \sum_{k=1}^{T-1} \rho v_i^*(k) + \phi(x_i^*(T), y_i^*(T)), \\ &\leq J^*(i) - \phi(x_i^*(0), y_i^*(0)) - \rho v_i^*(0) + \phi(x_i^*(T), y_i^*(T)), \\ J^*(q(i+1)) - J^*(q(i)) &\leq \phi(x_i^*(T), y_i^*(T)) - \phi(x_i^*(0), y_i^*(0)) - \rho v_i^*(0). \end{aligned} \tag{6.31}$$

Supstitucijom (6.20) u (6.31) slijedi

$$J^*(q(i+1)) - J^*(q(i)) \leq -\rho v_i^*(0), \tag{6.32}$$

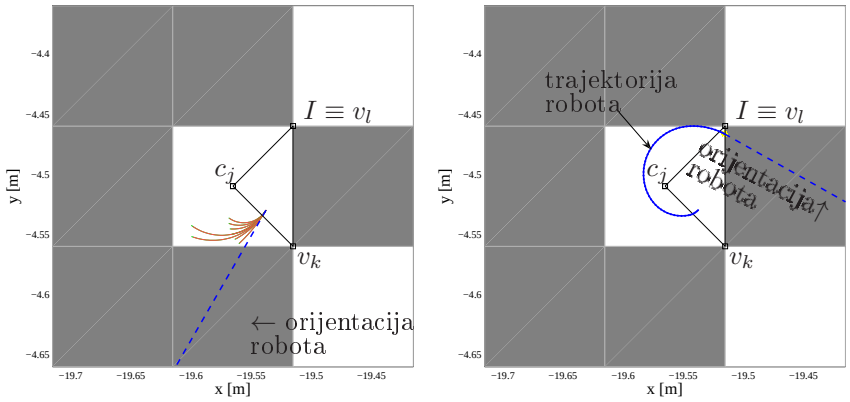
s jednakošću samo kada je $v_i^*(0) = 0$. Najmanja vrijednost $v_i^*(0) > 0$ određena je diskretizacijom skupa $\mathcal{V}_d$, koju označavamo s v_{min}. Budući da je skup $\mathcal{Q}$ ograničen postoji $R > 0$ takav da je $\mathcal{Q} \subset \{q \in \mathbb{R}^3 \mid \|q - q_G\| < R\}$. Prema tome može se odabrati kontinuirana funkcija $\gamma : [0, \infty\rangle \to [0, \infty\rangle$, $\gamma(t) > 0$ za sve $t > 0$ kao primjerice $\gamma(t) = \frac{v_{min}}{R} \cdot t$, za koju vrijedi

$$J^*(q(i+1)) - J^*(q(i)) \leq -\rho v_i^*(0) \leq -\gamma(\|q(i) - q_G\|). \tag{6.33}$$

Prema tome, izraz (6.5) je zadovoljen za $v_i^*(0) > 0$.

Potrebno je još pokazati da će se robot zaustaviti na putanji do cilja konačan broj puta, a pri svakom zaustavljanju izvesti odgovarajuću proceduru kojom se vrijednost J^* ne povećava.

Razmotrimo slučaj u kojem je trenutni položaj robota $q(i)$ unutar polja kojem je centar c_j, $j \in \mathcal{N}$ i sve ostvarive upravljačke sekvence imaju $v_i(0) = 0$ i proizvoljnu vrijednost $\omega_i(0) \in \mathcal{V}_d$. Takva je situacija prikazana na slici 6.8, lijevo. Robot je usmjeren prema prepreci u smjeru porasta cijene putanje i niti jedna prikazana trajektorija ne zadovoljava (6.20). Da bi se realiziralo kretanje prema cilju, robot se mora zakretati prema slobodnom prostoru i gibati u smjeru padajuće vrijednosti ϕ sve dok ne napusti kritično područje. Ovdje se uvodi ključna procedura koja uključuje prvo zakretanje, a

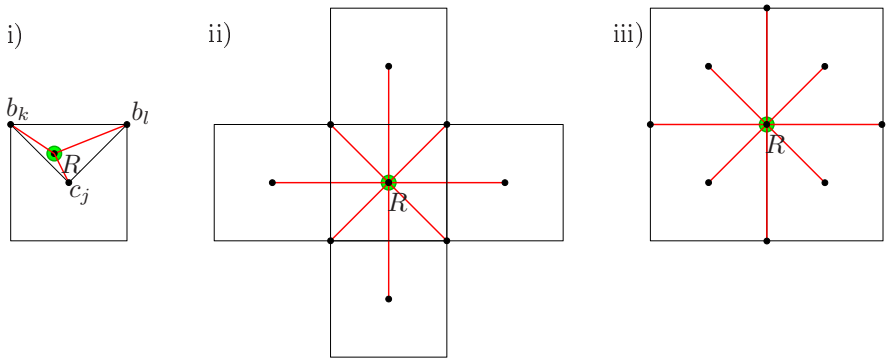


Slika 6.8. *Lijevo: robot je usmjeren prema prepreci i ne postoji ostvariva upravljačka sekvenca s* $v_i^*(0) \neq 0$. *Nijedna prikazana trajektorija ne zadovoljava* (6.20). *Desno: robot se počinje zakretati prema izlaznoj točki i čim se pojavi bolje rješenje počinje slijediti kružnu trajektoriju, što opet dovodi do slučaja* $v_i^*(0) = 0$, *a još nije stigao do izlazne točke.*

onda gibanje do točke minimalne vrijednosti ϕ dohvatljive iz trenutne pozicije gibanjem po pravocrtnoj putanji koja ne siječe niti jednu prepreku. Ta je točka nazvana izlaznom točkom, označenom s I.

Odvajanje ove procedure od optimizacije (6.14)-(6.20) i nakon toga nastavak optimizacije je opcionalno. Alternativno, može se provjeravati za vrijeme izvođenja procedure u svakom trenutku postoji li rješenje optimizacijskog problema (6.14)-(6.20) koji ima $v_i^*(0) \neq 0$ i primijeniti ovu upravljačku vrijednost. Međutim, moguće su situacije u kojima je alternativna metoda sporija od prve. Primjerice, situacija prikazana na slici 6.8, desno. Robot počinje zakretati prema izlaznoj točki i čim se pojavi bolje rješenje počinje slijediti kružnu trajektoriju, što opet zbog ograničenja gibanja dovodi do slučaja $v_i^*(0) = 0$, a još nije stigao do izlazne točke.

Izlazna je točka definirana s obzirom na položaj robota. Može biti vrh polja ili centar polja. Postoji opći slučaj definiranja izlazne točke (i), i dva specijalna slučaja (ii) i (iii) (slika 6.9).



Slika 6.9. *Određivanje izlazne točke.*

(i) Dva vrha polja najbliža točki $R(x(i), y(i))$ i centar polja (koji formiraju trokut unutar kojeg se nalazi točka R) su c_j, b_k, b_l. U općem je slučaju izlazna točka definirana kao jedna od te tri točke s najmanjom vrijednosti ϕ:

$$I = \arg\min\{\phi(c_j), \phi(b_k), \phi(b_l)\}. \tag{6.34}$$

Ako dvije točke imaju jednaku najmanju vrijednost, proizvoljno se odabire jedna od njih. Gibanjem do izlazne točke po pravocrtnoj putanji, vrijednost od ϕ se smanjuje zbog konstrukcije 2-simpleksa nad kojima je funkcija cijene putanje interpolirana. Zbog numeričkih izračuna pozicioniranje u izlaznoj točki unutar je male ε okoline od izlazne točke. Moguće su situacije u kojima je robot tik ispred izlazne točke, ali ju nikada ne dosiže jer ne može ostvariti proizvoljno mali pomak. Stoga su razmotreni specijalni slučajevi (ii) i (iii):

(ii) Ako je pozicija robota $R(x(i), y(i))$ unutar ε okoline od c_j, tada su razmotreni svi vrhovi od polja u kojem se nalazi R i svi centri susjednih polja koji imaju zajedničku stranicu s tim poljem:

$$\begin{aligned} I = \quad & \min_{n,b} \quad \{\phi(c_n), \phi(b)\} \\ & \text{u.u.} \quad n \in \mathcal{N} \text{ i } \|c_j - c_n\| = e_{polje}, \\ & \qquad b \in \mathcal{B} \text{ i } \|c_j - b\| = \frac{d_{polje}}{2}. \end{aligned} \tag{6.35}$$

(iii) Ako je pozicija robota $R(x(i), y(i))$ unutar ε okoline od vrha polja b_k, tada su razmotreni svi vrhovi polja koji leže na istoj stranici polja s b_k i svi centri polja kojima je zajednički taj vrh:

$$\begin{aligned} I = \quad & \min_{n,b} \quad \{\phi(c_n), \phi(b)\} \\ & \text{u.u.} \quad n \in \mathcal{N} \text{ i } \|b_k - c_n\| = \frac{d_{polje}}{2}, \\ & \qquad b \in \mathcal{B} \text{ i } \|b_k - b\| = e_{polje}. \end{aligned} \tag{6.36}$$

Gibanje do izlazne točke za slučajeve (ii) i (iii) uglavnom je po bridu trokuta (2-simpleksa) duž kojega se vrijednost ϕ smanjuje. Jedino u slučaju (ii), ako je izlazna točka u centru susjednog polja (od polja u kojem se $R(x(i), y(i))$ nalazi), gibanje je do izlazne točke po pravocrtnoj putanji koja prolazi kroz dva trokuta. Duž te linije vrijednost ϕ također se smanjuje jer vrijedi $\phi(I) < \phi(c_j)$. Ova nejednakost ne vrijedi jedino u slučaju ako je c_j ciljna koordinata, što znači da je robot dostigao ciljnu poziciju.

Nakon dolaska u izlaznu točku, ako ni dalje ne postoji $v_i^*(0) \neq 0$ u optimizacijskom problemu, robot mora zakretati i gibati se prema novoj izlaznoj točki. Izlazna točka može od pozicije robota biti udaljena najviše duljinu e_{polje}, a kut za koji se robot mora zakrenuti iznosi najviše 180°. Prema tome, procedura zakreta i dolaska u jednu izlaznu

točku završava u konačnom vremenu ograničenom odozgo

$$\begin{aligned} T_\Theta &\leq \frac{180^\circ}{\Delta t \cdot \omega_{min}}, \\ T_s &\leq \frac{e_{polje}}{\Delta t \cdot v_{min}}, \\ T_{izlaz} &= T_\Theta + T_s, \end{aligned} \tag{6.37}$$

gdje su $|\omega_{min}| \leq \Delta\omega_{max}$ i $|v_{min}| \leq \Delta v_{max}$ najmanje odabrane brzine iz diskretiziranog skupa brzina, različite od nule. Najmanji razmak dviju susjednih izlaznih točaka je $\frac{d_{polje}}{2}$. Duljina putanje koju bi robot prešao uzastopnim gibanjem od trenutne pozicije $(x(i), y(i))$ preko nove izlazne točke sve do cilja, usko je povezana s vrijednošću $\phi(x(i), y(i))$. Stoga je najveći broj zaustavljanja ograničen s

$$M = \lceil \frac{2\phi(x(i), y(i))}{d_{polje}} \rceil. \tag{6.38}$$

6.5 Eksperimentalni rezultati

Implementacija pomičnog horizonta testirana je u prostoru predstavljenim mrežastom kartom zauzeća. Iako je u dosadašnjim ilustracijama bio korišten diskretizirani skup brzina $\mathcal{V}_d$ s 35 parova (7 rotacijskih brzina i 5 translacijskih), to za izvođenje algoritma predstavlja zahtjevan vremenski izračun. U trenutnoj implementaciji iznosi do 50% trajanja vremenskog intervala (50 ms). Testiranja algoritma pokazuju vrlo dobre rezultate s najmanjim skupom $\mathcal{V}_d$ koji uključuje samo rubne i trenutne brzine, odnosno za devet parova brzina. Tada je srednje vrijeme izračuna, utrošeno na računanje svih ostvarivih trajektorija i optimizaciju funkcije J^*, 5 ms, a najveće vrijeme izračuna 14 ms. U testovima je zadana samo pozicija cilja, a orijentacija je proizvoljna. Zadana ciljna orijentacija u ovoj implementaciji ostvaruje se samo dodatnim zakretom na ciljnoj poziciji. Algoritam prekida izvođenje kada odabere $v^*(0) = 0$, a nalazi se unutar kruga pola polumjera robota od cilja. Dakle, ne čeka se završetak procedure zakreta i dolaska u izlaznu točku - cilj. U tablicama 6.1 i 6.2 dani su parametri koje koristi algoritam: najveća translacijska i rotacijska brzina v_{max} i ω_{max}, najveće promjene brzina Δv_{max} i $\Delta\omega_{max}$, trajanje jednog vremenskog intervala Δt, duljina horizonta T, najveći korišteni domet senzora S_{max}, broj upravljačkih vrijednosti u diskretiziranom skupu $\mathcal{V}_d$, polumjer robota r_r i širina polja e_{polje}. Izvedena su dva testa algoritma. Prvi je test

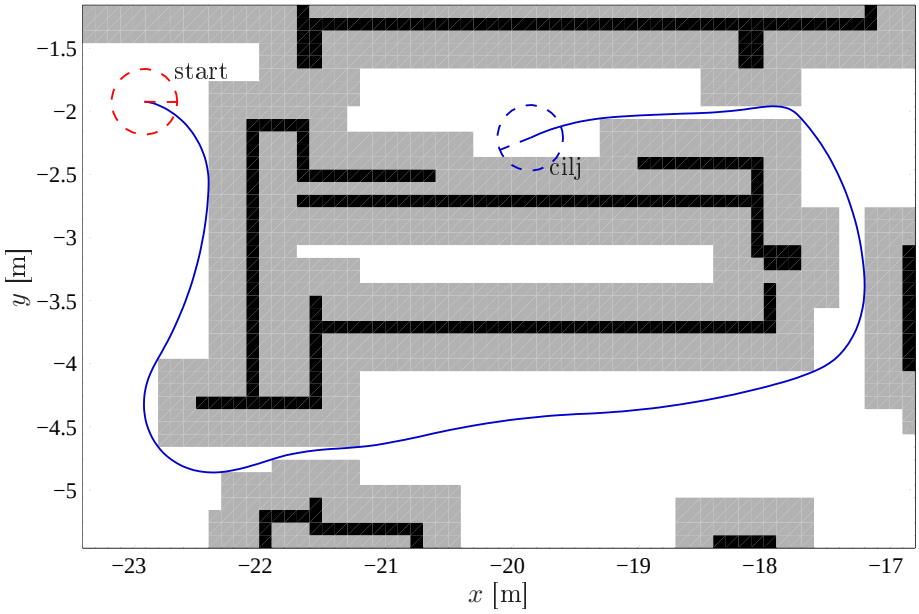
Tablica 6.1. *Kinematička i dinamička ograničenja robota*

v_{max} [mm/s]	ω_{max} [°/s]	Δv_{max} [mm/s]	$\Delta\omega_{max}$ [°/s]	Δt [s]
600	100	30	10	0.1

Tablica 6.2. *Vrijednosti parametara korištenih u implementaciji pomičnog horizonta*

T [intervali]	S_{max} [mm]	$\lvert\mathcal{V}_d\rvert$	r_r [mm]	e_{polje} [mm]
50	3000	9	260	100

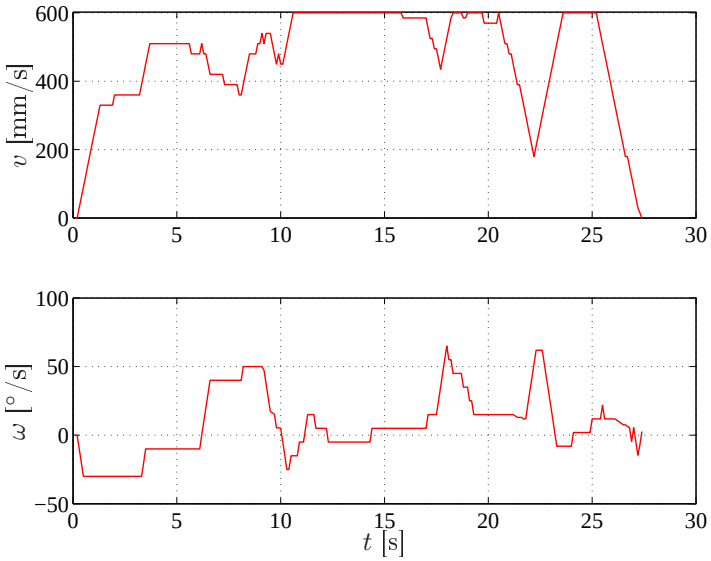
izveden u prostoru u kojem su prepreke gusto raspoređene, tako da na nekim mjestima robot ima slobodu gibanja unutar samo jednog polja, slika 6.10. Start i cilj odabrani su tako da se robot početno mora udaljavati od cilja. Sve su prepreke upisane u mrežastu kartu zauzeća i tijekom cijelog gibanja robota kroz prostor nema nikakvih promjena.



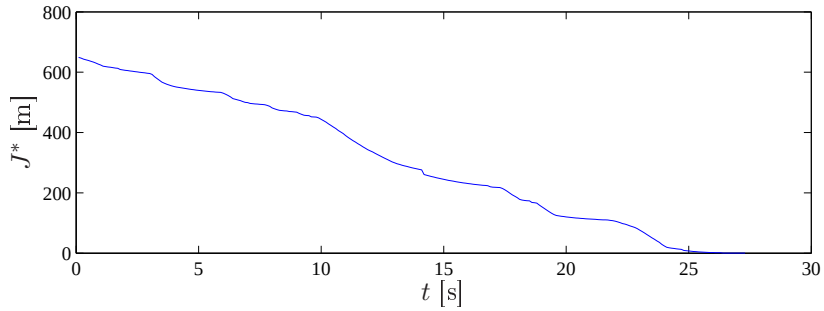
Slika 6.10. *Trajektorija robota u prvome testu.*

Odvožena je trajektorija robota glatka i niti na jednom mjestu ne prolazi kroz prepreku, što je osigurano postupkom opisanom u 6.3.1. Iako od algoritma nije zahtijevano točno pozicioniranje u cilju, završna točka trajektorije u ovom testu udaljena je 0.6 mm od pozicije cilja. Profil brzina dan je na slici 6.11. Najveća translacijska brzina ostvarena je u svim dijelovima prostora u kojima robot ne mora naglo skretati. Na slici 6.12 prikazane su vrijednosti funkcije J^* po svakoj točki trajektorije. Vidljivo je da vrijednosti funkcije J^* nikada ne rastu povećanjem vremenskog koraka. Na slici 6.13 prikazan je uvećani dio odvožene trajektorije. Strelice predstavljaju pokazivače algoritma D*, a kvadratići putanju koja je određena algoritmom D* od starta do cilja. Točke na trajektoriji označavaju poziciju robota u vremenskim trenucima proračuna upravljačkih vrijednosti. Gušće točke upućuju na sporije gibanje robota.

Drugi test izveden je u prostoru u kojem se nalaze tri skupine nepoznatih prepreka,

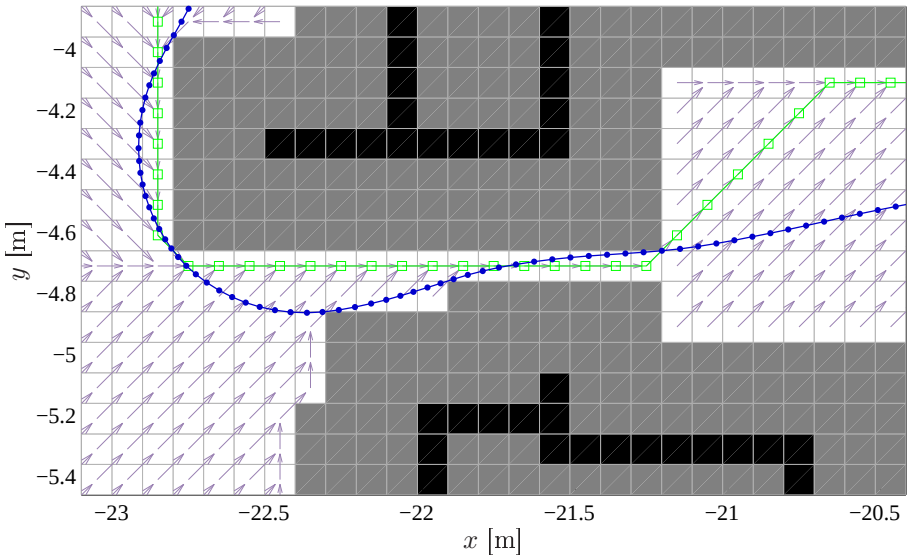


Slika 6.11. *Profil brzina robota u prvome testu.*

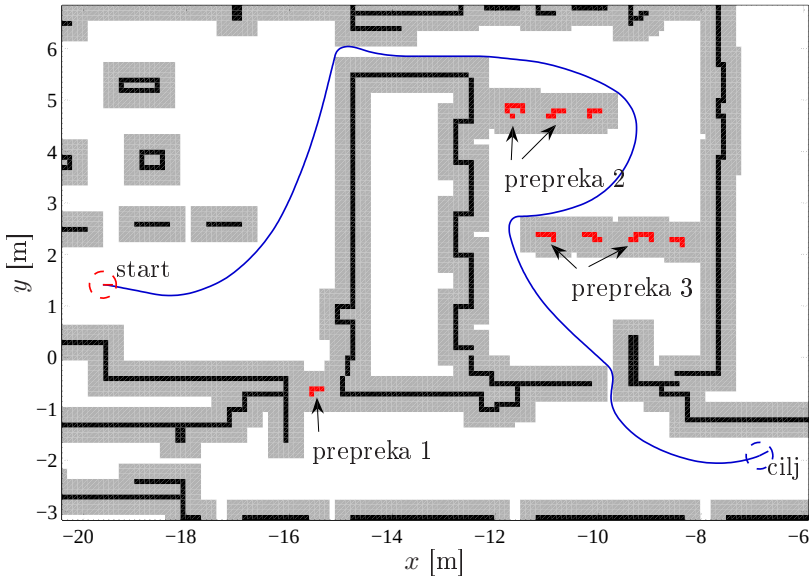


Slika 6.12. *Vrijednosti funkcije J^* u točkama trajektorije prvog testa.*

označene na slici 6.14. Inicijalna optimalna putanja prolazi kroz nepoznatu prepreku 1. Robot otkriva nepoznatu prepreku svojim senzorom kada joj se približi na udaljenost S_{max}. Informacije o novoj zauzetosti upisujuju se u mrežastu kartu zauzeća. Vrijednosti cijena g^* ažuriraju se minimalnom broju čvorova algoritmom D* i preusmjeravaju se pokazivači tako da tvore nove optimalne putanje iz svakog čvora do cilja. Nepoznata prepreka 2 nalazi se na novom optimalnoj putanji i ponovo se ažuriraju cijene g^* i pokazivači. Zadnja promjena u ovom testu je otkrivanje prepreke 3. Promjene cijena zbog nepoznatih prepreka vidljive su u skokovitim porastima vrijednosti funkcije J^* na slici 6.15. Nad svakim skokom u vrijednosti J^* naznačena je prepreka koja ga je uzrokovala. Nakon skoka u vrijednosti J^* u daljnjim koracima vrijednost J^* nikada ne raste (do novog skoka zbog promjene u prostoru). Za svaku promjenu u prostoru algoritam se ponaša kao da je u tom trenutku počeo računati, osim što mu početna brzina nije 0 nego neka trenutna. Profil brzina dan je na slici 6.16. Najveća moguća translacijska brzina ostvarena je u svim slobodnim dijelovima prostora udaljenijim od prepreka. Vidljivo

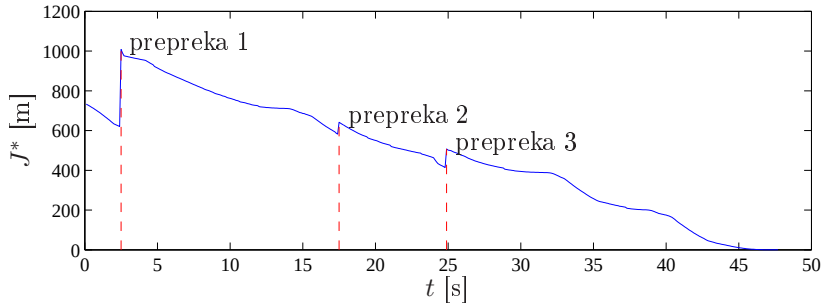


Slika 6.13. *Trajektorija robota u prvom testu (uvećano) s pokazivačima i putanjom algoritma D*.*

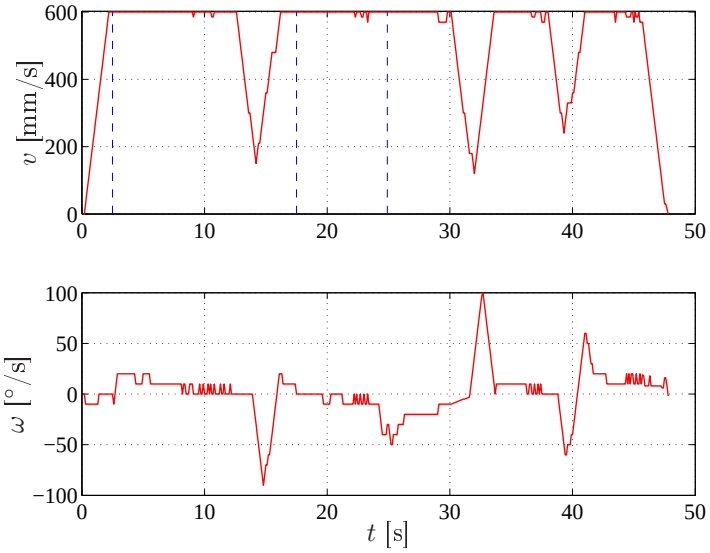


Slika 6.14. *Trajektorija robota u prostoru s tri skupine nepoznatih prepreka.*

je da je u svim trenucima replaniranja translacijska brzina ostala na najvećoj mogućoj vrijednosti. Na slici 6.17 prikazan je uvećani dio prostora u blizini prepreke 1 u trenutku prije otkrivanja prepreke. Ostvarive trajektorije prikazane su bojama prema parametru T_0, kao i na slici 6.4, a optimalna trajektorija prikazana je svjetloplavim



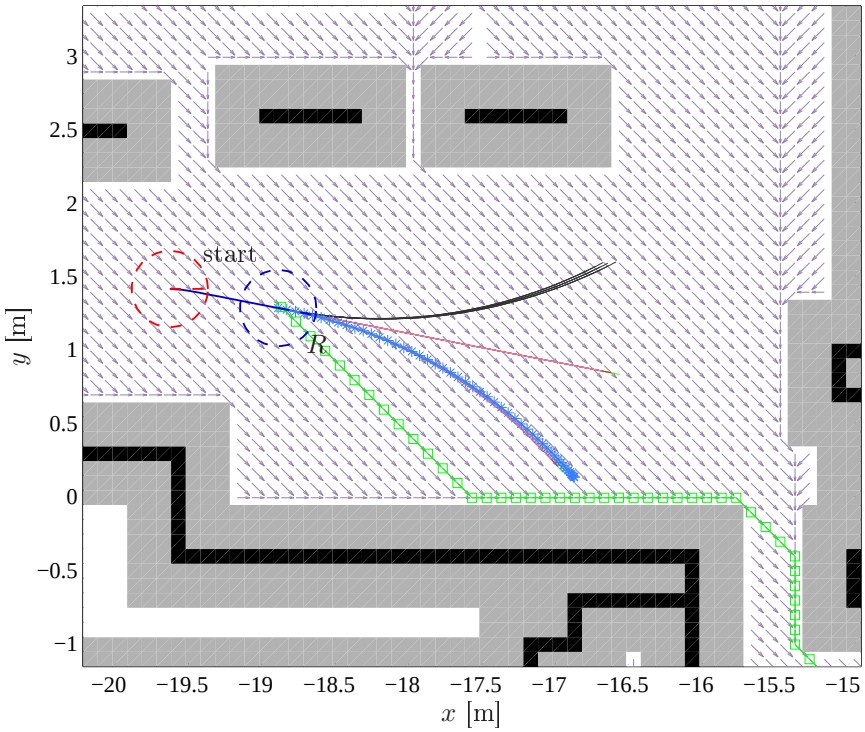
Slika 6.15. *Vrijednosti funkcije J^* u točkama trajektorije sa skokovima u trenucima replaniranja.*



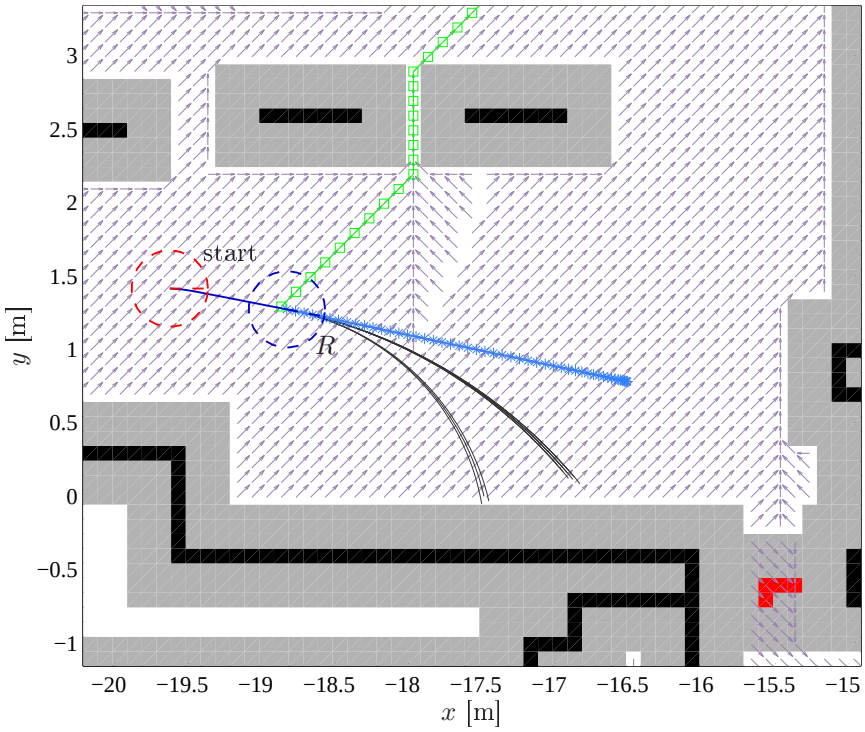
Slika 6.16. *Profil brzina robota u drugom testu s naznačenim trenucima replaniranja.*

zvjezdicama. Optimalna putanja prikazana je kvadratićima, a strelice predstavljaju pokazivače algoritma D*. Trenutni položaj robota označen je s R. U sljedećem je koraku robot otkrio prepreku 1 svojim senzorom, slika 6.18. Crne trajektorije predstavljaju one koje ne zadovoljavaju uvjet (6.20) zbog promjena cijena u prostoru. Valja primijetiti, iako nova putanja nakon replaniranja zbog prepreke 1 prolazi kroz uski prolaz između dviju prepreka, da robot odabire trajektoriju u skladu s ograničenjima, koja ima najveću brzinu i prolazi kroz širi slobodni prostor. Praćenje optimalne putanje zahtijevalo bi zaustavljanje, zakretanje u smjeru optimalne putanje i dalje nastavljanje gibanjem, što u ovom slučaju i prema kriterijskoj funkciji (6.14) nije optimalno rješenje. Ovaj test pokazuje da je algoritam efikasan i u dinamičkim prostorima. Odvožena je trajektorija glatka i zaobilazi nepoznate prepreke. Završna točka trajektorije udaljena je 0.4 mm od pozicije cilja.

Za uspješnost algoritma u dinamičkim prostorima mora vrijediti pretpostavka da



Slika 6.17. *Trajektorija robota u trenutku k, prije otkrivanja prepreke 1.*



Slika 6.18. *Trajektorija robota u trenutku $k+1$, nakon replaniranja zbog prepreke 1.*

se promjene događaju najviše jednom u vremenskom koraku tako da ih senzor može otkriti. Budući da algoritam D* ima sposobnost brzog replaniranja, prostor se može svaki vremenski korak smatrati statičnim. U dinamičkim prostorima nije moguće jamčiti konvergenciju gibanja do cilja u konačnom vremenu. Algoritam D* samo uzima u obzir moguće putanje u svakom koraku, i tako postoji mogućnost da zapne u alternirajućem otvaranju i zatvaranju mogućih putanja do cilja.

6.6 Sažetak

U ovom je poglavlju predložen algoritam planiranja gibanja pomoću upravljanja pomičnim horizontom. Predložena je kriterijska funkcija J^* koja objedinjuje algoritam dinamičkog prozora i algoritam D*. Kriterijska se funkcija J^* minimizira po unaprijed složenim ostvarivim trajektorijama dinamičkog prozora s obzirom na trenutno stanje robota i njegova ograničenja, a vrijednosti kriterijske funkcije računaju se prema vrijednostima navigacijske funkcije u točkama trajektorija dobivene interpolacijom cijena algoritma D*. U svakom diskretnom trenutku izvodi se optimizacija kriterijske funkcije J^* za T vremenskih intervala. Primjenjuje se samo prvi član proračunane optimalne upravljačke sekvence duljine T. Definiranjem ostvarivih trajektorija dinamičkog prozora s odgovarajućim profilom brzina, odnosno upravljačkim sekvencama, omogućeno je da u sljedećem koraku optimizacije postoji barem jedna ostvariva trajektorija koja će smanjiti iznos kriterijske funkcije. Za predloženi je algoritam planiranja gibanja dokazana asimptotska stabilnost ciljne točke korištenjem Ljapunovljeve analize stabilnosti, gdje je kriterijska funkcija J^* iskorištena kao Ljapunovljeva funkcija. Dodatno, određivanjem kontinuirane navigacijske funkcije jednostavnim postupkom interpolacije, omogućeno je da trajektorije konvergiraju pravom optimalnom rješenju. Kontinuirana navigacijska funkcija određuje se samo u točkama ostvarivih trajektorija robota što omogućuje izvršavanje algoritma u stvarnom vremenu i u slučaju promjena u prostoru. Eksperimentalni rezultati potvrđuju očekivane prednosti ovoga algoritma planiranja gibanja i za smanjen skup diskretnog prostora brzina. Algoritam pokazuje dobre rezultate i u dinamičkim prostorima pod pretpostavkom da su promjene spore dinamike.

Poglavlje 7

Zaključak

U ovom su radu istraženi algoritmi planiranja gibanja autonomnih mobilnih robota u dinamičkim i nepoznatim unutarnjim prostorima. Istraživanja su posebno bila usmjerena razvoju algoritama koji su prikladni za rad u stvarnom vremenu i koji osiguravaju učinkovito gibanje robota u velikim unutarnjim prostorima uz zajamčeno izbjegavanje gibajućih prepreka u prostoru. Problem planiranja gibanja razložen je na dva glavna potproblema: (1) planiranje geometrijskih putanja robota i (2) generiranje trajektorije gibanja robota radi osiguranja dobrog slijeđenja geometrijske putanje robota, poštujući njegova kinematička i dinamička ograničenja i raspored prepreka u njegovoj okolini.

U kontekstu planiranja geometrijskih putanja robota u poznatim prostorima, ali s mogućnošću pojave nepoznatih statičkih i dinamičkih prepreka u njima, istraženi su algoritmi zasnovani na pretraživanju mrežastih karata zauzeća. Posebno su istraženi algoritmi D* i FD* jer imaju svojstvo dinamičkog planiranja putanje, tj. jer omogućuju promjene putanje u stvarnome vremenu pri promjenama u prostoru. Međutim, algoritmi planiranja putanje zasnovani na mrežastim kartama zauzeća imaju dva značajna nedostatka: (1) dobivena je putanja sastavljena od niza povezanih linijskih segmenata s promjenama u orijentaciji segmenata koje su višekratnici od 45° i (2) računski su previše zahtjevni za planiranje putanje u velikim prostorima. Prvi se nedostatak odražava u potrebi za znatnim usporavanjem (često i zaustavljanjem) gibanja mobilnoga robota zbog skokovitih promjena orijentacije pri prelasku s praćenja jednog linijskog segmenta putanje na drugi. S ciljem rješavanja toga problema, u ovom je radu razvijen algoritam planiranja putanje, nazvan dvosmjernim algoritmom D*. Razvijeni algoritam izračunava najkraću putanju u geometrijskom prostoru na osnovi mrežaste karte zauzeća, pri čemu orijentacije segmenata putanje nisu višekratnici 45° već su proizvoljne. Algoritam nasljeđuje svojstvo dinamičkog planiranja putanje od algoritma D*. Razvijeni je algoritam provjeren eksperimentalno i uspoređen sa standardnim algoritmom D* pod istim uvjetima. Eksperimentima su potvrđene očekivane prednosti razvijenog algoritma planiranja putanje.

Problem planiranja putanje u velikim unutarnjim prostorima, primjerice prostorima

sastavljenima od više katova ili čak više zgrada, riješen je hijerarhijskom apstrakcijom prostora. Izvedena je nova hijerarhijska organizacija karte prostora kojom je osigurano optimalno horizontalno (u jednom katu) i vertikalno planiranje putanje (između katova zgrade). Razvijena je metoda automatske izgradnje takve hijerarhijske karte iz mrežaste karte zauzeća. Metoda jamči konzistentnost stvorene hijerarhije i omogućuje optimalno hijerarhijsko planiranje putanje u dinamičkim prostorima. Nadalje, razvijen je hijerarhijski algoritam planiranja putanje, nazvan FHD*, koji osigurava optimalnost putanje i poboljšava dinamičke karakteristike Cagigasova hijerarhijskog algoritma D*. Značajno su smanjeni vrijeme izračuna, cijena puta i utrošak memorije. Razvijena je strategija optimalnog postavljanja tzv. mosnih čvorova između kojih su izračunani predodređeni parcijalni putovi, čime je postignuto znatno ubrzanje procesa planiranja. Uspješnost algoritma FHD* usko je povezana s hijerarhijskom kartom, koja koristi znatno manje parcijalnih putova između mosnih čvorova od Cagigasova algoritma i time štedi memoriju. Eksperimentima je potvrđena očekivana prednost razvijenog algoritma FHD*.

Problem planiranja geometrijske putanje dodatno se usložnjava ako se mobilni robot mora autonomno gibati u nepoznatome prostoru, tj. u prostoru za koji mu karta nije dostupna. Taj se problem naziva problemom istraživanja prostora. S ciljem rješavanja toga problema razvijen je algoritam istraživanja poligonalnog prostora, kao proširenje Ekmanova algoritma istraživanja. Razvijenim su algoritmom istraživanja uklonjena stroga ograničenja izvornog Ekmanova algoritma koji pretpostavlja idealnu lokalizaciju robota i idealni senzor udaljenosti. Razvijeni algoritam nasljeđuje svojstvo konvergencije izvornog algoritma, a stvorena karta prostora objedinjuje utjecaj nesigurnosti senzorskih očitanja i položaja mobilnog robota, čime je postignuta konvergencija algoritma istraživanja u realnim uvjetima. Eksperimentalnim rezultatima na stvarnom robotu potvrđene su prednosti razvijenog algoritma istraživanja nad izvornim Ekmanovim algoritmom.

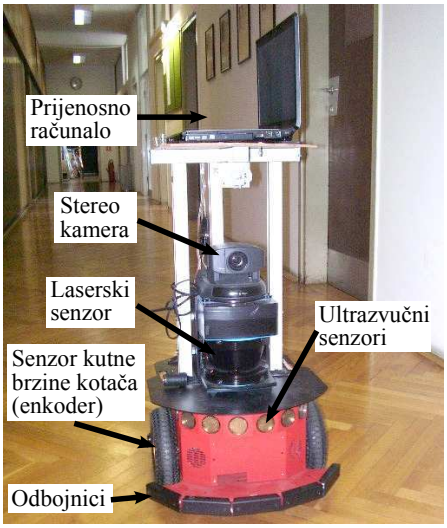
U kontekstu generiranja trajektorija gibanja istraženi su algoritmi slijeđenja putanje i izbjegavanja prepreka. Algoritmi izbjegavanja prepreka podložni su problemu lokalnog minimuma pa ih je potrebno kombinirati s algoritmom planiranja globalne geometrijske putanje. Opisane su metode slijeđenja isplanirane putanje zasnovane na algoritmu dinamičkog prozora koje proračunavaju moguće trajektorije s obzirom na kinematička i dinamička ograničenja robota. Uveden je kriterij objedinjavanja algoritma planiranja putanje i algoritma dinamičkog prozora, prema kojemu se uspoređuju trajektorije dinamičkog prozora s globalnom putanjom. Razvijenim objedinjenim algoritmom planiranja gibanja jamči se sigurno gibanje robota do globalnog cilja u prisustvu statičkih prepreka. Algoritam je zatim proširen tako da osigurava sigurno i glatko gibanje i među gibajućim preprekama, uz pretpostavku da najveće brzine gibajućih prepreka nisu veće od najveće brzine robota. Gibanje prepreka predstavljeno je gibanjem zauzetih gibajućih polja u mrežastoj karti zauzeća. Predviđene trajektorije svakog gibajućeg polja korištene su u procjeni točke sudara s trajektorijama dinamičkog prozora. Eksperimentima je potvrđena učinkovitost razvijenog objedinjenog algoritma planiranja gibanja. Dodatno,

ostvareno je učinkovito gibanje više robota koji su upravljani proširenim objedinjenim algoritmom bez uvođenja prometnih pravila i uz pretpostavku da svaki robot zna zadnju odluku drugih robota. Nadalje, da bi prošireni objedinjeni algoritam planiranja gibanja bio primjenjiv u dugotrajnom gibanju robota kroz prostor popunjen mnogim gibajućim preprekama uvedena je strategija koja rješava problem zastoja u uskim prolazima. Zastoj se javlja u situacijama kada se robot i neka gibajuća prepreka (npr. drugi robot ili čovjek) sretnu u uskom prolazu i jedno drugom onemoguće nastavak gibanja prema cilju. Uz pretpostavku najstrožeg uvjeta, tj. da robot s gibajućim preprekama ne može komunicirati već njihovo ponašanje opaža samo svojim percepcijskim senzorima, razvijena je strategija sprječavanja zastoja zasnovana na konceptu sprječavanja zastoja u ethernet mrežama. Uspješnost razvijene strategije provjerena je eksperimentima na stvarnom robotu.

U završnom je poglavlju rada opisan integrirani algoritam planiranja gibanja mobilnog robota koji osigurava konvergenciju gibanja robota prema zadanoj ciljnoj poziciji. Razvijeni se algoritam planiranja gibanja zasniva na konceptu pomičnog horizonta. Uvedena je takva kriterijska funkcija koja objedinjuje unaprijed složene trajektorije dinamičkog prozora i navigacijsku funkciju izvedenu iz cijena algoritma D*. Uvedenom kriterijskom funkcijom osigurano je da u sljedećem koraku optimizacije postoji barem jedna ostvariva trajektorija koja će smanjiti iznos kriterijske funkcije. Za razvijeni je algoritam planiranja gibanja dokazana asimptotska stabilnost ciljne točke korištenjem Ljapunovljeve analize stabilnosti. Eksperimentima su potvrđena očekivana svojstva ovoga algoritma planiranja gibanja, kao i mogućnost njegova izvršavanja u stvarnom vremenu u slučaju promjena u prostoru.

DODATAK A

Opis mobilnog robota PIONEER 3-DX



Slika A.1. *Mobilni robot PIONEER 3-DX.*

Mobilna platforma PIONEER 3-DX korištena u svim eksperimentima prikazana je na slici A.1. Kretanje omogućuju dva pogonska kotača, koje pokreću dva nezavisna elektromotora. Zahvaljujući tome ovakav robot posjeduje svojstvo holonomičnosti, tj. mogućnost okretanja u mjestu. Treći kotač je kastorski kotač, koji osigurava statičku stabilnost. Maksimalna linearna brzina robota je 1.6 m/s. Na osovine pogonskih kotača ugrađeni su enkoderi. Oni omogućuju određivanje prijeđenog puta na temelju podataka o zakretu osovine kotača. Zaštitu pri sudaru robotu pružaju odbojnici, tj. kontaktni prekidači koji signaliziraju dodir tijela robota s okolinom. Ultrazvučni senzori – sonari raspoređeni su u dva polukružna polja smještena na prednjoj i stražnjoj strani platforme. Iznad sonarskih polja montiran je laserski senzor udaljenosti (engl. *laser range finder*, LRF), koji se vrlo dobro može nadopunjavati sa sonarima za potrebe npr. lo-

kalizacije i izbjegavanja prepreka. Njegove prednosti su vrlo visoka preciznost i kutna razlučivost, znatno veći domet od sonara, te vrlo velika brzina rada i vrlo mali broj lažnih očitanja. S druge strane nedostatci su mu što pokriva samo vrlo tanku plohu u prostoru i ne može detektirati optički prozirne objekte. Robot je također opremljen i elektroničkim kompasom, čiji je osnovni problem osjetljivost na vanjske smetnje. Pogonski su motori i senzori (osim laserskog senzora udaljenosti i kamera) upravljani Siemens-ovim C166 mikrokontrolerom s P2OS ugradbenim operacijskim sustavom za rad u stvarnom vremenu. Sve upravljačke akcije više razine odvijaju se na prijenosnom računalu postavljenom na robotsku platformu, pod Linux operacijskim sustavom.

Literatura

[1] R. Agrawal and HV Jagadish. Materialization and incremental update of path information. In *Data Engineering, 1989. Proceedings. Fifth International Conference on*, pages 374–383, 1989.

[2] K.O. Arras, J. Persson, N. Tomatis, and R. Siegwart. Real-Time Obstacle Avoidance for Polygonal Robots with a Reduced Dynamic Window. *ICRA'02, IEEE International Conference on Robotics and Automation*, 2002.

[3] C. Backstrom and P. Jonsson. Planning with abstraction hierarchies can be exponentially less efficient. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI-95*, 1995.

[4] T. Bandyopadhyay, Z. Liu, M.H. Ang, and W.K.G. Seah. Visibility-based exploration in unknown environment containing structured obstacles. In *Proceedings of the 12th International Conference on Advanced Robotics (ICAR)*, pages 484–491, 2005.

[5] T. Belker and D. Schulz. Local action planning for mobile robot collision avoidance. In *IEEE/RSJ International Conference on Intelligent Robots and System, 2002*, volume 1, 2002.

[6] S.J. Bence, K.F. Riley, and M.P. Hobson. *Mathematical Methods for Physics and Engineering: A Comprehensive Guide*. Cambridge: Cambridge University Press, 2006.

[7] M. Bennewitz, W. Burgard, and S. Thrun. Finding and optimizing solvable priority schemes for decoupled path planning techniques for teams of mobile robots. *Robotics and Autonomous Systems*, 41(2-3):89–99, 2002.

[8] C. Bordenave, D. Mcdonald, and A. Proutière. Random multi-access algorithms: A mean field analysis. In *Proceedings of Allerton conference*. Citeseer, 2005.

[9] J. Borenstein, H. R. Everett, and R. Feng. *Where am I? Sensors and Methods for Mobile Robot Positioning*. University of Michigan, Ann Arbor, MI 48109, 1996.

[10] J. Borenstein and Y. Koren. The Vector Field Histogram - Fast Obstacle Avoidance for Mobile Robots. *IEEE Transactions on Robotics and Automation*, 7(3):278–288, 1991.

[11] G.A. Borges and M.J. Aldon. A split-and-merge segmentation algorithm for line extraction in 2Drange images. In *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, volume 1, 2000.

[12] M. Brezak. *Localization, motion plannig and control of mobile robots in intelligent spaces*. PhD thesis, University of Zagreb, 2010.

[13] O. Brock and O. Khatib. High-speed navigation using the Global Dynamic Window Approach. *ICRA'99, IEEE International Conference on Robotics and Automation*, 1999.

[14] R. Brooks. Visual map making for a mobile robot. In *1985 IEEE International Conference on Robotics and Automation. Proceedings*, volume 2, pages 824–829, 1985.

[15] SJ Buckley, I.B.M.T.J.W.R. Center, and Y. Heights. Fast motion planning for multiple moving robots. In *1989 IEEE International Conference on Robotics and Automation, 1989. Proceedings.*, pages 322–326, 1989.

[16] A. Bundy, F. Giunchiglia, R. Sebastiani, and T. Walsh. Calculating criticalities. *Artificial Intelligence*, 88(1-2):39–67, 1996.

[17] W. Burgard, A.B. Cremers, D. Fox, D. Hähnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun. Experiences with an interactive museum tour-guide robot. *Artificial Intelligence*, 114(1-2), 2000.

[18] D. Cagigas. Hierarchical D* algorithm with materialization of costs for robot path planning. *Robotics and Autonomous Systems*, 52(2-3):190–208, 2005.

[19] D. Cagigas and J. Abascal. Hierarchical Path Search with Partial Materialization of Costs for a Smart Wheelchair. *Journal of Intelligent and Robotic Systems*, 39(4):409–431, 2004.

[20] D. Castro, U. Nunes, and A. Ruano. Reactive local navigation. In *IEEE 28th Annual Conference of the Industrial Electronics Society (IECON)*, volume 3, pages 2427–2432. Citeseer, 2002.

[21] A. Chakravarthy and D. Ghose. Obstacle avoidance in a dynamic environment: A collision cone approach. *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*, 28(5):562–574, 1998.

[22] CC Chang and K.T. Song. Environment prediction for a mobile robot in a dynamic environment. *IEEE transactions on robotics and automation*, 13(6):862–872, 1997.

[23] C.H. Chiang, P.J. Chiang, J.C.C. Fei, and J.S. Liu. A Comparative Study of Implementing Fast Marching Method and A* Search for Mobile Robot Path Planning in Grid Environment: Effect of Map Resolution. In *IEEE Workshop on Advanced Robotics and its Social Impacts*, pages 7–12, 2007.

[24] DK Cho and M. Chung. Intelligent Motion Control Strategy for a Mobile Robot in the Presence of Moving Obstacles. In *Proceedings of the IEEE International Workshop on Intelligent Robots and Systems (IROS)*, volume 91, pages 541–546, 1991.

[25] EW Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[26] Borko Dimitrijević. Algoritam za sprječavanje sukoba mobilnog robota s drugim gibajućim objektima u uskim prolazima. *Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva, diplomski rad*, 2009.

[27] A. Ekman, A. Torne, and D. Stromberg. Exploration of polygonal environments using range data. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 27(2):250–255, 1997.

[28] M. Erdmann and T. Lozano-Perez. On multiple moving objects. *Algorithmica*, 2(1):477–521, 1987.

[29] H.J.S. Feder and J.J.E. Slotine. Real-time path planning using harmonic potentials in dynamic environments. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 874–881. INSTITUTE OF ELECTRICAL ENGINEERS INC (IEEE), 1997.

[30] D. Ferguson and A. Stentz. Multi-resolution Field D. *Intelligent Autonomous Systems 9: IAS-9*, 2006.

[31] D. Ferguson and A. Stentz. Field D*: An Interpolation-Based Path Planner and Replanner. *Robotics Research: Results of the 12 th International Symposium ISRR(STAR: Springer Tracts in Advanced Robotics Series Volume 28)*, 28:239–253, 2007.

[32] JL Fernandez, R. Sanz, JA Benayas, and AR Dieguez. Improving collision avoidance for mobile robots in partially known environments: the beam curvature method. *Robotics and Autonomous Systems*, 46(4):205–219, 2004.

[33] Juan-Antonio Fernández-Madrigal and Javier González. Hierarchical graph search for mobile robot path planning. *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, 1:656–661, 1998.

[34] Juan-Antonio Fernández-Madrigal and Javier González. Multihierarchical graph search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(1):103–113, 2002.

[35] P. Fiorini and Z. Shiller. Motion planning in dynamic environments using velocity obstacles. *The International Journal of Robotics Research*, 17(7):760–772, 1998.

[36] D. Fox, W. Burgard, and S. Thrun. The dynamic window approach to collision avoidance. *Robotics & Automation Magazine, IEEE*, 4(1):23–33, 1997.

[37] K. Fujimura and H. Samet. A hierarchical strategy for path planning among moving obstacles [mobile robot]. *IEEE Transactions on Robotics and Automation*, 5(1):61–69, 1989.

[38] C. Galindo, Juan-Antonio Fernández-Madrigal, and J. González. Improving efficiency in mobile robot task planning through world abstraction. *Robotics, IEEE Transactions on [see also Robotics and Automation, IEEE Transactions on]*, 20(4):677–690, 2004.

[39] R. Gartshore and P. Palmer. Exploration of an unknown 2D environment using a view improvement strategy. *Towards Autonomous Robotic Systems*, pages 57–64, 2005.

[40] SS Ge and YJ Cui. Dynamic motion planning for mobile robots using potential field method. *Autonomous Robots*, 13(3):207–222, 2002.

[41] D. B. Gennery. Traversability analysis and path planning for a planetary rover. *Autonomous Robots*, 6(2):131–146, 1999.

[42] E. Giunchiglia and T. Walsh. Using abstraction. In *Proc. of the 8th Conference of the Society for the Study of Artificial Intelligence and Simulation of Behaviour*, pages 225–234, 1990.

[43] F. Giunchiglia. Using Abstrips Abstractions–Where do We Stand? *Artificial Intelligence Review*, 13(3):201–213, 1999.

[44] Graham C. Goodwin, María M. Seron, and José A. De Doná. *Constrained control and estimation: an optimisation approach*. Springer Verlag, 2004.

[45] C.E. Guestrin. *Planning under uncertainty in complex structured environments*. PhD thesis, STANFORD UNIVERSITY, 2003.

[46] E.A. Hansen and R. Zhou. Anytime heuristic search. *Journal of Artificial Intelligence Research*, 28(1):267–297, 2007.

[47] E.A. Hansen, S. Zilberstein, and V.A. Danilchenko. Anytime heuristic search: First results. *Tech. rep. 97-50, Univ. of Massachusetts/Amherst, Dept. of Computer Science*, 1997.

[48] P.E. Hart, N.J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[49] J. Hershberger and S. Suri. An optimal algorithm for Euclidean shortest paths in the plane. *SIAM Journal on Computing*, 28(6):2215–2256, 1999.

[50] Y.W. Huang, N. Jing, and EA Rundensteiner. A Semi-Materialized View Approach for Route Guidance in Intelligent Vehicle Highway Systems. In *Pro. Second ACM Workshop Geographic Information System*, pages 144–151. Citeseer, 1994.

[51] Y.W. Huang, N. Jing, and E.A. Rundensteiner. A Hierarchical Path View Model for Path Finding in Intelligent Transportation Systems. *GeoInformatica*, 1(2):125–159, 1997.

[52] K.S. Hwang and M.Y. Ju. A propagating interface model strategy for global trajectory planning among moving obstacles. *IEEE Transactions on Industrial Electronics*, 49(6):1313–1322, 2002.

[53] Šandor Ileš. Snimanje i vektorizacija karte mobilnog robota uz iscrpno modeliranje šuma te spajanje s postojećim algoritmom istraživanja prostora. *Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva, diplomski rad*, 2009.

[54] Šandor Ileš, Marija Seder, and Ivan Petrović. Improvement of map building during the exploration of polygonal environments using the range data. In *Proceedings of the International Conference on Electrical Drives and Power Electronics (EDPE 2009)*, 2009.

[55] J. Illingworth and J. Kittler. A survey of the hough transform. *Computer Vision, Graphics, and Image Processing*, 44(1):87 – 116, 1988.

[56] N. Jing, Y.W. Huang, and E.A. Rundensteiner. Hierarchical optimization of optimal path finding for transportation applications. In *Proceedings of the fifth international conference on Information and knowledge management*, pages 261–268. ACM New York, NY, USA, 1996.

[57] Srećko Jurić-Kavelj, Marija Seder, and Ivan Petrović. Tracking multiple moving objects using adaptive sample-based joint probabilistic data association filter. In *Proceedings of 5th International Conference on Computational Intelligence, Robotics and Autonomous Systems (CIRAS 2008)*, pages 99–104, 2008.

[58] Ivan Kašić. Planiranje putanje mobilnog robota za autonomno pretraživanje nepoznatog unutarnjeg prostora. *Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva, diplomski rad*, 2008.

[59] O. Khatib. Real-Time Obstacle Avoidance for Manipulators and Mobile Robots. *The International Journal of Robotics Research*, 5(1):90–98, 1986.

[60] J.-O. Kim and P. Khosla. Real-time obstacle avoidance using harmonic potential functions. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, volume 1, pages 790–796, 1991.

[61] R. Kimmel, N. Kiryati, and A.M. Bruckstein. Multivalued distance maps for motion planning on surfaces with moving obstacles. *IEEE Transactions on Robotics and Automation*, 14(3):427–436, 1998.

[62] C.A. Knoblock. Learning abstraction hierarchies for problem solving. *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 923–928, 1990.

[63] N.Y. Ko and B.H. Lee. Avoidability measure in moving obstacle avoidance problem and itsuse for robot motion planning. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 3, pages 1296–1303, 1996.

[64] S. Koenig and M. Likhachev. Fast replanning for navigation in unknown terrain. *IEEE Transactions on Robotics*, 21(3):354–363, 2005.

[65] K. Konolige. A gradient method for realtime robot control. In *Proc. of the IEEE/RSJ Interational Conference on Intelligent Robotic Systems (IROS)*, pages 639–646. Citeseer, 2000.

[66] R.E. Korf. Planning as search: a quantitative approach. *Artificial Intelligence*, 33(1):65–68, 1987.

[67] D.L. Kreher and D.R. Stinson. *Combinatorial algorithms: generation, enumeration, and search*. CRC, 1999.

[68] J.C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Dodrecht, Netherlands, 1991.

[69] S.M. LaValle. *Planning algorithms*. Cambridge Univ Pr, 2006.

[70] J.H. Lee, B.H. Lee, and M.H. Choi. A real-time traffic control scheme of multiple AGV systems for collision free minimum time motion: a routing table approach. *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*, 28(3):347–358, 1998.

[71] Y. Li and K. Gupta. Motion planning of multiple agents in virtual environments on parallel architectures. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 1009–1014. Citeseer, 2007.

[72] M. Likhachev, G. Gordon, and S. Thrun. ARA*: Anytime A* with provable bounds on sub-optimality. *Advances in Neural Information Processing Systems*, 16, 2004.

[73] D.K. Liu, X. Wu, A.K. Kulatunga, and G. Dissanayake. Motion coordination of multiple autonomous vehicles in dynamic and strictly constrained environments. In *2006 IEEE Conference on Cybernetics and Intelligent Systems*, pages 1–6, 2006.

[74] S. Liu, L. Mao, and J. Yu. Path Planning Based on Ant Colony Algorithm and Distributed Local Navigation for Multi-Robot Systems. In *Proceedings of the 2006 IEEE International Conference on Mechatronics and Automation*, pages 1733–1738, 2006.

[75] T. Lozano-Pérez. Spatial planning: A configuration space approach. *IEEE Transactions on Computers*, C-32(2):108–120, 1983.

[76] VJ Lumelsky, S. Mukhopadhyay, and K. Sun. Dynamic path planning in sensor-based terrain acquisition. *IEEE Transactions on Robotics and Automation*, 6(4):462–472, 1990.

[77] V.J. Lumelsky and A.A. Stepanov. Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape. *Algorithmica*, 2(1):403–430, 1987.

[78] M.J. Mataric. Integration of representation into goal-driven behavior-based robots. *IEEE Transactions on robotics and automation*, 8(3):304–312, 1992.

[79] J. Matas, C. Galambos, and J. Kittler. Progressive probabilistic hough transform. In *Proc British Machine Vision Conference BMVC98*, pages 256–265, 1998.

[80] W.T. McCormick Jr, P.J. Schweitzer, and T.W. White. Problem decomposition and data reorganization by a clustering technique. *Operations Research*, 20(5):993–1009, 1972.

[81] Patricia Mellodge and Pushkin Kachroo. *Model abstraction in dynamical systems: application to mobile robot control*. Springer Verlag, 2008.

[82] J. Minguez and L. Montano. Nearness Diagram Navigation (ND): A New Real Time Collision Avoidance Approach. *IROS'00, IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2000.

[83] J.S.B. Mitchell. Geometric shortest paths and network optimization. *Handbook of computational geometry*, 411, 1998.

[84] Petar Mostarac. Planiranje putanja gibanja mobilnog robota u velikim unutarnjim prostorima primjenom hijerarhijskog D* algoritma. *Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva, diplomski rad*, 2007.

[85] Y.S. Nam, B.H. Lee, and NK Ko. An analytic approach to moving obstacle avoidance using an artificial potential field. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS*, volume 2, pages 482–487, 1995.

[86] P. Ogren and N.E. Leonard. A convergent dynamic window approach to obstacle avoidance. *IEEE Transactions on Robotics*, 21(2):188–195, 2005.

[87] C.S. Oliver, M. Saptharishi, J. Dolan, A. Trebi-Ollennu, and P. Khosla. Multi-robot path planning by predicting structure in a dynamic environment. In *Proceedings of the First IFAC Conference on Mechatronic Systems*, volume 2, pages 593–598. Citeseer, 2000.

[88] B. Oommen, S. Iyengar, N. Rao, and R. Kashyap. Robot navigation in unknown terrains using learned visibility graphs. part i: The disjoint convex obstacle case. *Robotics and Automation, IEEE Journal of*, 3(6):672–681, December 1987.

[89] I.P. Park and J.R. Kender. Topological direction-giving and visual navigation in large environments. *Artificial intelligence*, 78(1-2):355–395, 1995.

[90] L.E. Parker. Path Planning and Motion Coordination in Multiple Mobile Robot Teams. *Encyclopedia of Complexity and System Science*, 2009.

[91] Judea Pearl and Jin H. Kim. Studies in Semi-Admissible Heuristics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 4(4):392–399, 1982.

[92] S.T. Pfister. Weighted line fitting and merging. *California Institute of Technology, Tech. Rep*, 2002.

[93] S.T. Pfister. *Algorithms for mobile robot localization and mapping, incorporating detailed noise modeling and multi-scale feature extraction.* PhD thesis, California Institute of Technology, 2006.

[94] S.T. Pfister, SI Roumeliotis, and JW Burdick. Weighted line fitting algorithms for mobile robot map building and efficient data representation. In *IEEE International Conference on Robotics and Automation, 2003. Proceedings. ICRA'03*, volume 1, 2003.

[95] R. Philippsen. *Motion planning and obstacle avoidance for mobile robots in highly cluttered dynamic environments.* PhD thesis, ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE, 2004.

[96] R. Philippsen and R. Siegwart. An interpolated dynamic navigation function. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation (ICRA)*, volume 4, pages 3782–3789. Citeseer, 2005.

[97] R. Philipsen and R. Siegwart. Smooth and efficient obstacle avoidance for a tour guide robot, in 'IEEE Int. In *Conf. on Robotics and Automation', Taipei, Taiwan*, 2003.

[98] E. Pink and Q. Yang. A spectrum of plan justifications. In *Proc. AAAI Spring Symp. (AAAI-93)*, pages 29–33. Citeseer, 1993.

[99] P. Pirjanian and H.I. Christensen. Behavior coordination using multiple-objective decision making. In *Sensor Fusion and Decentralized Control in Autonomous Robotic Systems*, volume 3209, pages 78–89. Citeseer, 1997.

[100] P. Pirjanian and M. Matarić. A decision-theoretic approach to fuzzy behavior coordination. pages 101 –106, 1999.

[101] I. Pohl. Heuristic search viewed as path finding in a graph. *Artificial Intelligence*, 1(3-4):193–204, 1970.

[102] S. Quinlan. *Real-time modification of collision-free paths.* PhD thesis, Citeseer, 1994.

[103] S. Quinlan and O. Khatib. Elastic bands: Connecting path planning and control. In *In Proceedings of the International Conference on Robotics and Automation (ICRA)*, volume 2, pages 802–807, 1993.

[104] P.F. Reynolds Jr, A. Natrajan, and S. Srinivasan. Consistency maintenance in multiresolution simulation. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 7(3):392, 1997.

[105] E. Rimon and D.E. Koditschek. Exact robot navigation using artificial potential functions. *IEEE Transactions on robotics and automation*, 8(5):501–518, 1992.

[106] S.I. Roumeliotis and G.A. Bekey. SEGMENTS: a layered, dual-Kalman filter algorithm for indoorfeature extraction. In *2000 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2000.(IROS 2000). Proceedings*, volume 1, 2000.

[107] S.J. Russell, P. Norvig, J.F. Canny, J. Malik, and D.D. Edwards. *Artificial intelligence: a modern approach*. Prentice Hall Englewood Cliffs, NJ, 1995.

[108] D. Sack and W. Burgard. A comparison of methods for line extraction from range data. In *Proceedings of the 5th IFAC Symposium on Intelligent Autonomous Vehicles (IAV)*. Citeseer, 2004.

[109] J.R. Sack and J. Urrutia. *Handbook of computational geometry*. North-Holland, 2000.

[110] H. Samet. Neighbor Finding Techniques for Images Represented by Quadtrees. *Computer Graphics and Image Processing*, 18:37–57, 1982.

[111] C. Schlegel. Fast local obstacle avoidance under kinematic and dynamicconstraints for a mobile robot. In *1998 IEEE/RSJ International Conference on Intelligent Robots and Systems, 1998. Proceedings.*, volume 1, 1998.

[112] Marija Seder, Srećko Jurić-Kavelj, and Ivan Petrović. Automatic Creation of Hierarchical Maps for Indoor Environments. In *Proceedings of the Fifth International Conference on Computational Intelligence, Robotics and Autonomous Systems (CIRAS 2008)*, pages 19–24, 2008.

[113] Marija Seder, Kristijan Maček, and Ivan Petrović. An integrated approach to real-time mobile robot control in partially known indoor environments. In *Proceedings of the 31st Annual Conference of the IEEE Industrial Electronics Society (IECON 2005)*, pages 1785–1790, 2005.

[114] Marija Seder, Petar Mostarac, and Ivan Petrović. Hierarchical path planning of mobile robots in complex indoor environments. *Transactions of the Institute of Measurement and Control, doi:10.1177/0142331208100107*, 2009.

[115] Marija Seder and Ivan Petrović. Dynamic window based approach to mobile robot motion control in the presence of moving obstacles. In *Proceedings of the 2007 IEEE International Conference on Robotics and Automation (ICRA 2007)*, pages 1986–1991, 2007.

[116] Marija Seder and Ivan Petrović. Integration of Focused D* and Witkowski's algorithm for path planning and replanning. In *Proceedings of the 4th European Conference on Mobile Robots (ECMR 2009)*, pages 99–104, 2009.

[117] Marija Seder, Ivan Petrović, and Kristijan Maček. Motion Control of Mobile Robots in Indoor Dynamic Environments. In *Proceedings of the International Conference on Electrical Drives and Power Electronics (EDPE 2005)*, 2005.

[118] JA Sethian. A fast marching level set method for monotonically advancing fronts. *Proceedings of the National Academy of Sciences*, 93(4):1591–1595, 1996.

[119] A. Siadat, A. Kaske, S. Klausmann, M. Dufaut, and R. Husson. An optimized segmentation method for a 2D laser-scanner applied to mobile robot navigation. In *Proceedings of the 3rd IFAC symposium on intelligent components and instruments for control applications*, 1997.

[120] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo. *Robotics: modelling, planning and control*. Springer Verlag, 2008.

[121] R. Simmons. The Curvature-Velocity Method for Local Obstacle Avoidance. *ICRA'96, IEEE International Conference on Robotics and Automation*, 1996.

[122] R. Simmons. The Lane-Curvature Method for Local Obstacle Avoidance. *IROS'98, IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1998.

[123] M. Sniedovich. Dijkstra's algorithm revisited: the dynamic programming connexion. *Control and cybernetics*, 35(3):599, 2006.

[124] R. Spence and S. Hutchinson. An integrated architecture for robot motion planning and control inthe presence of obstacles with unknown trajectories. *IEEE Transactions on Systems, Man and Cybernetics*, 25(1):100–110, 1995.

[125] C. Stachniss and W. Burgard. An Integrated Approach to Goal-directed Obstacle Avoidance under Dynamic Constraints for Dynamic Environments. *IROS'02, IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2002.

[126] A. Stentz. Optimal and efficient path planning for partially-known environments. *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*, pages 3310–3317, 1994.

[127] A. Stentz. The focussed D* algorithm for real-time replanning. *Proceedings of the International Joint Conference on Artificial Intelligence*, 8, 1995.

[128] J.M.R.S. Tavares and A.J. Padilha. A new approach for merging edge line segments. In *RecPad'95-7th Portuguese Conference on Pattern Recognition*, 1995.

[129] S. Thrun, M. Beetz, M. Bennewitz, W. Burgard, AB Cremers, F. Dellaert, D. Fox, D. Hahnel, C. Rosenberg, N. Roy, et al. Probabilistic algorithms and the interactive museum tour-guide robot minerva. *The International Journal of Robotics Research*, 19(11):972, 2000.

[130] S. Thrun, W. Burgard, and D. Fox. *Probabilistic robotics*. Cambridge, Massachusetts: MIT Press, 2005.

[131] B. Tovar, R. Murrieta-Cid, and SM LaValle. Distance-optimal navigation in an unknown environment without sensing distances. *IEEE Transactions on Robotics*, 23(3):506–518, 2007.

[132] J.P. van den Berg. *Path Planning in Dynamic Environments*. PhD thesis, Utrecht University, 2007.

[133] B.R. Vatti. A generic solution to polygon clipping. *Communications of the ACM*, 35(7):56–63, 1992.

[134] CM Witkowski. A parallel processor algorithm for robot route planning. *Int. Joint Conf. on Artificial Intelligence (IJCAI), Karlsruhe, West Germany*, pages 827–829, 1983.

[135] D. Wooden and M. Egerstedt. Oriented visibility graphs: low-complexity planning in real-time environments. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2354–2359, 2006.

[136] L. Xu, L.G. Zhang, D.G. Chen, and Y.Z. Chen. The mobile robot navigation in dynamic environment. *Proceedings of the Sixth International Conference on Machine Learning and Cybernetics*, pages 566–571, 2007.

[137] M. Yamamoto, M. Shimada, and A. Mohri. On-Line Navigation of Mobile Robot Under the Existence of Dynamically Moving Multiple 0bstacles. In *Proceedings of the 4th IEEE International Symposium on Assembly and Task Planning*, pages 13–18, 2001.

[138] B. Yamauchi, A. Schultz, and W. Adams. Integrating exploration and localization for mobile robots. *Adaptive Behavior*, 7(2):217, 1999.

Sažetak

U ovom je radu istražen problem planiranja gibanja autonomnih mobilnih robota u dinamičkim i nepoznatim unutarnjim prostorima s posebnim naglaskom na algoritme prikladne za rad u stvarnom vremenu koji osiguravaju učinkovito gibanje robota uz zajamčeno izbjegavanje gibajućih prepreka u prostoru. Razvijen je algoritam planiranja geometrijske putanje zasnovan na mrežastoj karti zauzeća koji izračunava najkraću putanju u geometrijskom prostoru. Razvijen je algoritam izgradnje hijerarhijske karte prostora te algoritam hijerarhijskog planiranja putanje čime je osigurano planiranje gibanja mobilnog robota u stvarnom vremenu u velikim unutarnjim prostorima. Problem planiranja gibanja mobilnog robota u nepoznatim prostorima riješen je razvijenim algoritmom istraživanja prostora kojime je postignuta konvergencija algoritma istraživanja u realnim uvjetima. Opisane su metode slijeđenja isplanirane putanje, zasnovane na algoritmu dinamičkog prozora, koje proračunavaju moguće trajektorije s obzirom na kinematička i dinamička ograničenja robota. Uveden je kriterij objedinjavanja algoritma planiranja putanje i algoritma dinamičkog prozora. Razvijenim objedinjenim algoritmom planiranja gibanja osigurano je sigurno i glatko gibanje bez zastoja među statičkim i gibajućim preprekama. Naposlijetku, razvijen je algoritam planiranja gibanja zasnovan na konceptu pomičnog horizonta, za koji je dokazana asimptotska stabilnost ciljne točke korištenjem Ljapunovljeve analize stabilnosti. Svi su predloženi algoritmi testirani simulacijski i eksperimentalno na stvarnom robotu.

Ključne riječi:

mobilni robot, planiranje putanje, hijerarhijska apstrakcija prostora, hijerarhijsko planiranje, istraživanje prostora, planiranje gibanja, izbjegavanje prepreka, izbjegavanje gibajućih prepreka

Abstract

Motion planning of autonomous mobile robots in dynamic and unknown indoor environments

This thesis focuses on the problem of path planning for autonomous mobile robots in dynamic and unknown indoor environment. Special emphasis is set on algorithms suitable for real-time applications which should ensure the efficient movement of a robot with guaranteed moving obstacles avoidance in the environment. A path planning algorithm based on the grid map that calculates the shortest path in the geometric space is proposed. An algorithm of automatic creation of the hierarchical graph is proposed together with the hierarchical path planning algorithm suitable for mobile robot's motion in real-time in a complex indoor environment. The problem of motion planning of mobile robot in unknown environment is solved by the proposed exploration algorithm, by which convergence of the exploration task in real terms is achieved. The path following methods based on dynamic window approach are considered, which take into account kinematic and dynamic constraints of the robot. The criterion for integration of the path planning algorithm and the dynamic window algorithm is introduced. The integrated motion planning algorithm is proposed which ensures a safe and smooth motion among moving obstacles. Finally, the motion planning algorithm based on the concept of receding horizon control is proposed, for which asymptotic stability of the goal point is proven via Lyapunov stability analysis. All proposed algorithms are tested by simulation and experimentally on a real robot in laboratory environments.

Keywords:

mobile robot, path planning, hierarchical abstraction of space, hierarchical planning, exploration, motion planning, obstacle avoidance, moving obstacles avoidance

Životopis

Marija Đakulović (rođena Seder) rođena je 18. svibnja 1981. u Zagrebu. Maturirala je 1999. godine u prirodoslovno matematičkoj V. Gimnaziji u Zagrebu. Studij na Fakultetu elektrotehnike i računarstva u Zagrebu započela je 1999. godine. Godine 2001. upisala je smjer automatika, te joj je 2002. Fakultetsko vijeće odobrilo završetak studija s naglaskom na znanstvenoistraživačkom radu pod mentorstvom prof.dr.sc Ivana Petrovića u području mobilne robotike. Dobitnica je nagrade Fakulteta Josip Lončar za osobit uspjeh na drugoj i četvrtoj godini dodiplomskog studija i rektorove nagrade za studentski rad na petoj godini dodiplomskog studija. Diplomirala je u svibnju 2004. godine. Od srpnja 2004. godine radi na Zavodu za automatiku i računalno inženjerstvo Fakulteta elektrotehnike i računarstva kao znanstveni novak. Glavno područje njenog znanstvenog interesa je mobilna robotika (posebno planiranje putanje, izbjegavanje prepreka i planiranje gibanja). Kao asistent sudjeluje u nastavi na Fakultetu iz predmetâ Automatsko upravljanje, Računalno upravljanje sustavima i Laboratorij i vještine Matlab. Održava tečajeve za programiranje programirljivih logičkih kontrolera tvrtke Siemens.

Curriculum Vitae

Marija Đakulović (born Seder) was born on May 18, 1981 in Zagreb, Republic of Croatia. She graduated in 1999 at the High school of natural sciences and mathematics in Zagreb. She enroled the Faculty of Electrical Engineering and Computing in 1999, where in 2001 she took her specialization in the field of automatic control. In 2002 the Faculty council accepted her study completion with emphasis on scientific research under the supervison of prof. Ivan Petrović in the field of mobile robotics. She received the Josip Lončar award for outstanding results in the second and fourth year of undergraduate study, and the Vice-chancellor award for student work in the fifth year of undergraduate study. She received the M.Sc. degree in Electrical Engineering from the Faculty of Electrical Engineering and Computing (FER Zagreb), University of Zagreb, Croatia in May, 2004. In the same year in July she joined the Department of Control and Computer Engineering at FER Zagreb as a research assistant. Her main research interest are in the field of mobile robotics (especially path planning, obstacle avoidance and motion planning). She is a teaching assistant in the following undergraduate courses: Automatic Control, Computer-Controlled Systems and Laboratory and Skills - Matlab. As a part of an ongoing FER and Siemens collaboration she holds courses on programmable logic controllers and their application.