Planiranje putanje mobilnog robota s teretom nepravilnog oblika

Rusković, Ana

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 102

PLANIRANJE PUTANJE MOBILNOG ROBOTA S TERETOM NEPRAVILNOG OBLIKA

Ana Rusković

Zagreb, veljača 2025.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 102

PLANIRANJE PUTANJE MOBILNOG ROBOTA S TERETOM NEPRAVILNOG OBLIKA

Ana Rusković

Zagreb, veljača 2025.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 30. rujna 2024.

DIPLOMSKI ZADATAK br. 102

Pristupnica:	**Ana Rusković (0036521769)**
Studij:	Informacijska i komunikacijska tehnologija
Profil:	Automatika i robotika
Mentorica:	prof. dr. sc. Marija Seder

Zadatak: **Planiranje putanje mobilnog robota s teretom nepravilnog oblika**

Opis zadatka:

Mobilni roboti često se koriste za prijevoz različitih vrsta tereta velikih dimenzija unutar skladišta ili postrojenja. Ograničen radni prostor, uski prolazi i nepravilan oblik tereta koji prelazi gabarite robota predstavljaju izazov kod planiranja putanje mobilnih robota i zahtijevaju precizno manevriranje teretom u radnoj okolini ljudi i strojeva. U ovom diplomskom radu potrebno je razviti algoritam za planiranje putanje mobilnog robota zasnovan na A* algoritmu uzimajući u obzir pretraživanje diskretiziranog prostora pozicija i orijentacija ovisno o obliku robota. Potrebno je implementirati algoritam planiranja putanje robota u prostorima u kojima dolazi do izražaja potreba za preciznim određivanjem orijentacije robota prilikom prolaska kroz uske prolaze kako bi se zadržala mogućnost prolaska kroz iste. Razvijeni algoritam potrebno je testirati na različitim simulacijskim scenarijima.

Rok za predaju rada: 14. veljače 2025.

Zahvala

Prije svega, želim zahvaliti mentorici prof. dr. sc. Mariji Seder i asistentici mag. ing. Jeleni Penavi na nesebičnoj pomoći, stručnim savjetima i trudu kojeg su uložile vodeći me kroz ovaj rad i studij.

Posebnu zahvalu dugujem svojoj obitelji, roditeljima Mariji i Nevenu i sestri Andrei na neizmjernoj podršci i motivaciji tijekom cijelog školovanja i što su uvijek bili uz mene. Također, hvala teti Jadranki koja je podržavala svaki moj korak.

Hvala svim prijateljima koji su sa mnom bili dio ovog putovanja i učinili ga nezaboravnim.

Na kraju, zahvalna sam sebi što nikada nisam odustala.

Sretno i uspješno!

Sadržaj

1. Uvod

Robot je programabilni stroj koji integrira hardver i softver, a dizajniran je za obavljanje specifičnih zadataka autonomno ili poluautonomno. Roboti mogu varirati u složenosti, od jednostavnih robota koji obavljaju ponavljajuće zadatke u kontroliranim okruženjima do naprednih sustava sposobnih za donošenje odluka, učenje i prilagođavanje dinamičnim, nepredvidivim uvjetima. Raspon primjena stoga je širok, od industrijske automatizacije i proizvodnje do istraživanja, uslužnih i servisnih zadataka. Stvoreni su kako bi replicirali ili proširili ljudske sposobnosti, a pritom smanjili ljudske pogreške, poboljšavajući učinkovitost i povećavajući sigurnost.

Mobilni roboti bitna su podgrupa robota, dizajnirana za kretanje i interakciju s okolinom dok obavlja zadatke poput automatizacije skladišta, autonomnog transporta ili misije potrage i spašavanja. Opremljeni su senzorima, aktuatorima, kontrolnim sustavima i sofisticiranim algoritmima za navigaciju koji im omogućuju opažanje okoline, obradu informacija i izvođenje fizičkih radnji. Planiranje putanje temeljni je izazov u području mobilne robotike, osobito kada roboti imaju zadatak upravljati u složenim okruženjima pritom noseći nepravilan teret. Sposobnost robota da učinkovito izračuna optimalnu putanju od početne do ciljne pozicije pritom izbjegavajući prepreke i uzimajući u obzir varijacije tereta, ključna je za njegovu izvedbu u aplikacijama u stvarnom svijetu. Tradicionalni algoritmi za planiranje puta, kao što je najčešće korišteni A* algoritam, pružaju korisnu osnovu za navigaciju u strukturiranim okruženjima. Međutim, često su neadekvatni kada opterećenje robota značajno utječe na njegovu dinamiku kretanja i orijentaciju, uvodeći dodatna ograničenja koja se moraju uzeti u obzir u procesu planiranja putanje.

Ovaj diplomski rad usmjeren je na razvoj prilagođene verzije A* algoritma koja uključuje ograničenja orijentacije i opterećenja mobilnih robota, pridonoseći pouzdanijoj navigaciji u složenim okruženjima. Klasični A* algoritam proširuje se integracijom orijentacije kao ključnog parametra, čime robot uzima u obzir svoju fizičku konfiguraciju i raspodjelu opterećenja pri odabiru putanje. Ova je modifikacija posebno važna za robote koji nose nepravilan teret, gdje promjene u orijentaciji mogu utjecati na stabilnost, upravljanje i energetsku učinkovitost. Uzimajući u obzir te čimbenike, prilagođenim A* algoritmom cilj je generirati realistične i učinkovite putanje, poboljšavajući ukupnu izvedbu robota. Prilagođeni A* algoritam definiran u ovom radu razvijan je pomoću programskog okruženja Matlab u kojem je simulirana i praktična primjena algoritma s različitim oblicima tereta i prostornih prepreka.

2. Radni i konfiguracijski prostor mobilnog robota

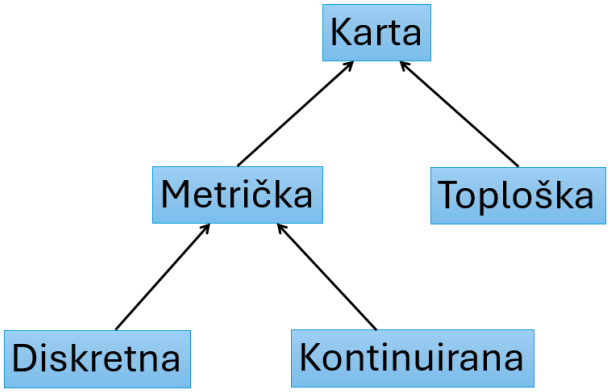
Upotreba mobilnih robota u današnjem svijetu sve je češća pojava u svim sferama života, od uslužnih djelatnosti i medicine do industrije i prijevoza. Definiranje prostora u kojem se robot nalazi i kreće, omogućuje uspješno planiranje putanje i interakciju robota s okolinom. Kako bi se robot kretao u prostoru, potrebno je odrediti njegov položaj u prostoru, odnosno svakoj točki robota mora se točno odrediti pozicija u prostoru kako ne bi došlo do sudara s preprekama. Prostor u kojem se robot nalazi, može se opisati kao radni ili konfiguracijski prostor. Radni prostor opisuje fizičko okruženje u kojem robot djeluje, dok konfiguracijski prostor obuhvaća sve moguće konfiguracije, odnosno položaje koje robot može zauzeti, s obzirom na njegove stupnjeve slobode i ograničenja.

2.1. Radni prostor

Radni je prostor prostor u koji robot može doći ili istražiti ga, a opisan je kartezijskim koordinatama (npr. x, y, z za 3D okruženja ili x, y za 2D okruženja). Radni prostor mobilnog robota podijeljen je na slobodni i zauzeti prostor. Slobodan prostor dio je okoline u kojem se robot može kretati bez nailaženja na prepreke, dok zauzeti prostor predstavlja prepreke koje onemogućuju kretanje robota, poput zidova, namještaja ili drugih fizičkih objekata. U većini slučajeva mobilni roboti kreću se unutar dinamičnih okruženja, gdje se radni prostor može mijenjati tijekom vremena zbog kretanja ljudi, vozila ili drugih prepreka. Razumijevanje radnog prostora kao dinamičnog omogućuje robotu kontinuirano prilagođavanje promjenama u okolini i izbjegavanje sudara.

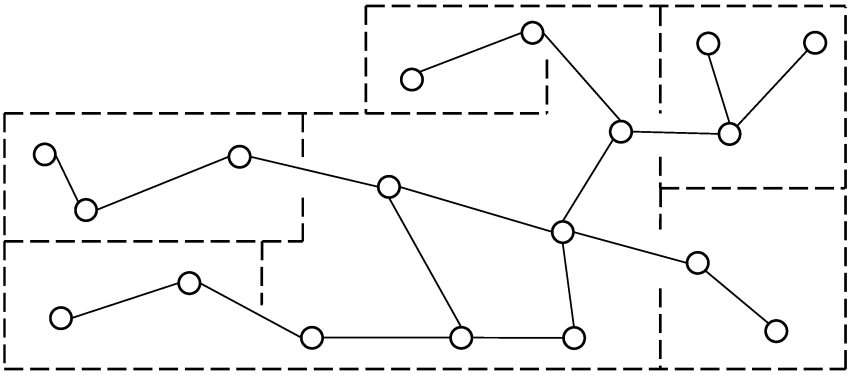
2.2. Vrste prikaza radnog prostora

Kako bi robot uspješno planirao putanju kretanja i uopće se kretao u prostoru, potreban je prikaz prostora koji može nastati prije početka kretanja robota ili nastaje tijekom njegova kretanja. Neovisno o načinu nastanka, prostor u kojem se robot kreće možemo prikazati kao topološku kartu ili kontinuiranu ili diskretnu metričku kartu[2]. Navedena podjela prikazana je na slici 2.1.



Slika 2.1. Različiti prikazi prostora

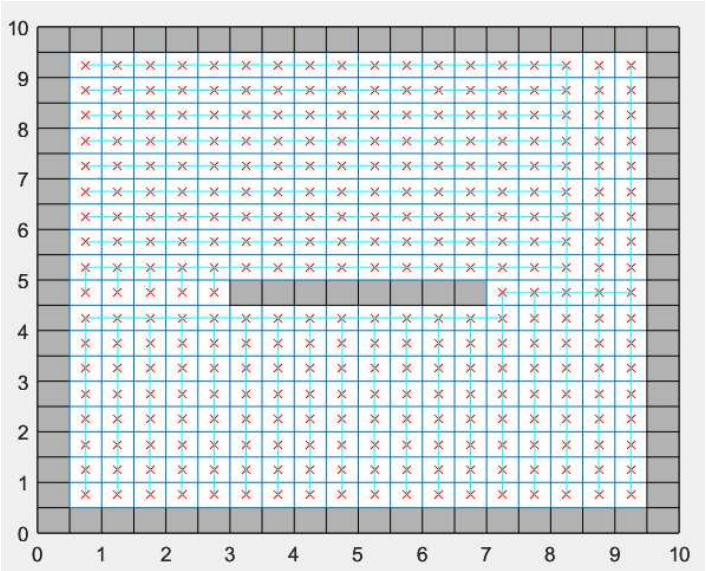
Topološka karta apstraktno i relativno prikazuje prostor korištenjem čvorova i bridova. Čvorovi predstavljaju određene pozicije u prostoru koje su međusobno povezane bridovima koji, ovisno o karakteristikama povezanosti, nose određenu cijenu. Bridovi između pojedinih čvorova predstavljaju staze ili rute između navedenih čvorova, što ukazuje na to da postoji izravna veza (hodnik, cesta ili prolaz) između dviju pozicija. Odnos između čvorova temelji se na susjedstvu i pristupačnosti, a ne na preciznim prostornim koordinatama. Topološke karte naglašavaju povezanost i relativni položaj, a primjer takve karte dan je na slici 2.2. One su kompaktnije od metričkih jer ne pohranjuju detaljne geometrijske informacije. To ih čini računalno učinkovitima i idealnima za sustave s ograničenom memorijom ili procesorskom snagom. Usredotočujući se samo na ključne pozicije i njihove veze, topološke karte smanjuju složenost okoliša, a istovremeno pružaju dovoljno informacija za navigaciju. Jedan od glavnih nedostataka topoloških karata jest nepostojanje detaljnih geometrijskih ili metričkih informacija. U okruženjima gdje je potrebno precizno kretanje (kao što su uski hodnici ili područja s gustim preprekama), nedostatak točnih udaljenosti i kutova može otežati navigaciju.



Slika 2.2. Prikaz topološke karte [3]

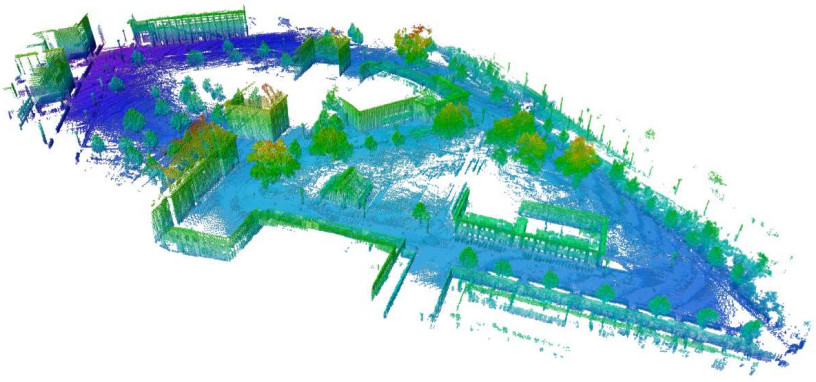
Metrička karta najčešće je korištena. Ona prikazuje okolinu pružajući precizne geometrijske informacije o lokacijama, udaljenostima i prostornim odnosima između objekata. Metričke karte omogućuju autonomnim sustavima da razumiju svoju okolinu i da se točno pozicioniraju unutar iste. Metričke karte mogu se klasificirati u dvije kategorije: diskretne i kontinuirane.

Diskretne metričke karte dijele prostor u na diskretne ćelije ili jedinice, pri čemu svaka jedinica odgovara određenoj poziciji i sadrži informaciju o prisutnosti prepreka. Rezolucija diskretne metričke karte ovisi o rezoluciji mreže, pri čemu veća rezolucija dopušta preciznije detalje o prostoru. Najčešće korištena struktura diskretne metričke karte jest mrežasta karta zauzeća koja je prikazana na slici 2.3. U navedenoj strukturi, okolina je podijeljena na mrežu kvadratnih ili pravokutnih ćelija, a svakoj je ćeliji dodijeljena vrijednost na temelju toga je li odgovarajući prostor slobodan ili zauzet. Zauzete ćelije, koje su predstavljene vrijednošću 1 kod binarnih karata zauzeća, označavaju ćelije koje sadrže prepreku, a slobodne ćelije, predstavljene vrijednošću 0, označavaju prostor koji je prazan, odnosno slobodan za kretanje. Diskretne metričke karte jednostavne su za implementaciju i upotrebu u mnogim algoritmima za planiranje putanje. Određivanje zauzetosti prostora jednostavan je proces provjere vrijednosti odgovarajuće ćelije u mreži. Omogućuju skaliranje na različite rezolucije što dovodi do kompromisa između rezolucije i učinkovitosti. Karte visoke rezolucije pružaju više detalja, ali zahtijevaju više resursa, dok su karte niske rezolucije računalno učinkovite, ali manje precizne.



Slika 2.3. Prikaz diskretne metričke karte

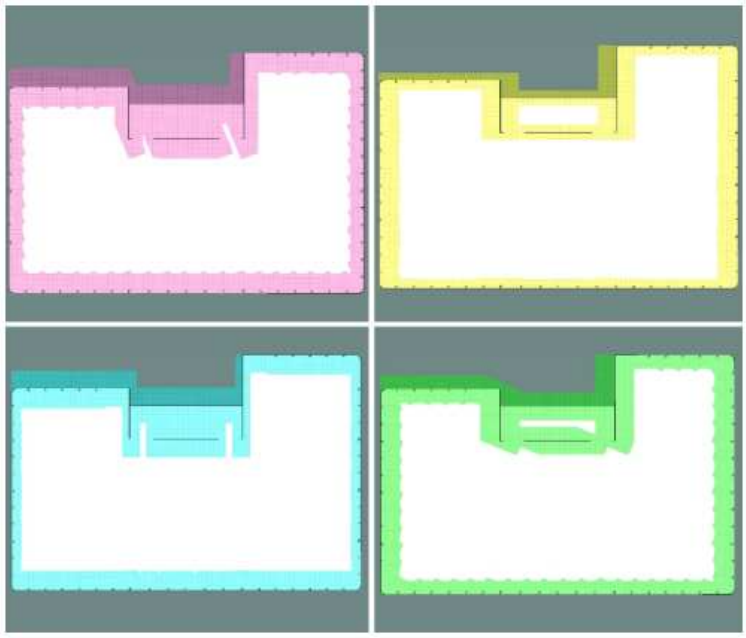
Kontinuirane metričke karte pružaju kontinuiran opis prostora uz visoki stupanj preciznosti i detalja. Za razliku od aproksimacija svojstvenih diskretnim kartama, kontinuirane metričke karte imaju izravan, neprekinut pogled na prostor, što omogućuje sustavima mjerenje preciznih udaljenosti između točaka i snimanje finih detalja okoline. Podatci prikupljeni sa senzora poput lidara, kamera i drugih tehnologija za otkrivanje dubine, kreiraju kartu koja precizno i metrički prikazuje geometriju stvarnog svijeta, a pohranjuju se kao geometrijski modeli, oblaci točaka ili matematičke funkcije. Na slici 2.4. prikazana je kontinuirana metrička karta pohranjena kao oblak točaka. Zbog toga su navedene karte složenije za generiranje, pohranjivanje i obradu, što zahtijeva značajne računalne resurse. S obzirom na to da se kontinuirane karte ne oslanjaju na diskretne ćelije, roboti mogu glatko upravljati prostorom bez trzaja ili prilagodbi putanje do kojih može doći prilikom prijelaza između ćelija mreže u diskretnim kartama.



Slika 2.4. Prikaz kontinuirane metričke karte [4]

2.3. Konfiguracijski prostor

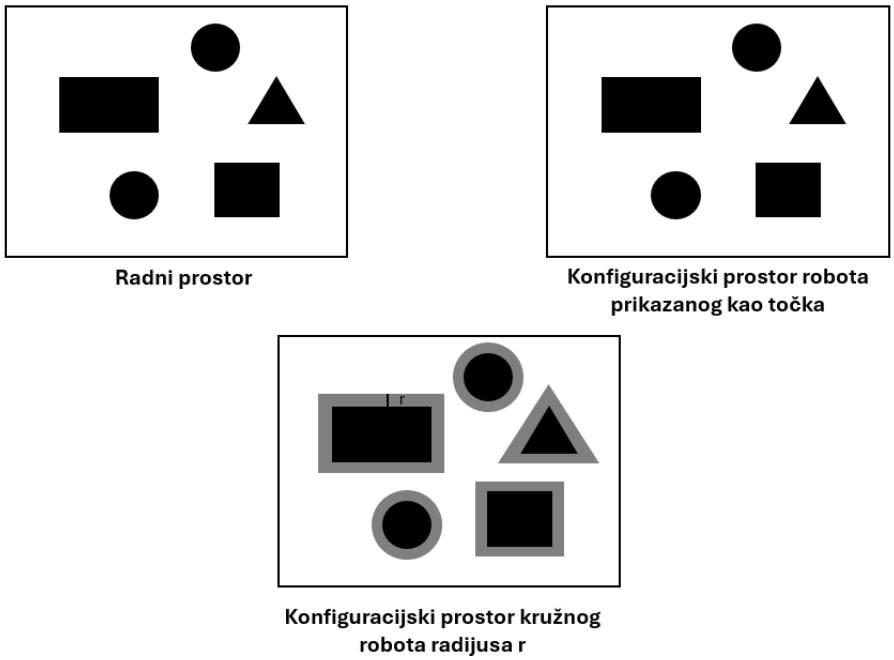
Konfiguracijski prostor ili C-prostor robota odnosi se na skup svih mogućih konfiguracija koje robot može zauzeti, s obzirom na njegov položaj, orijentaciju i ograničenja kretanja. Konfiguracija robotskog sustava potpuna je specifikacija položaja svake točke tog sustava [5]. Za mobilne robote koji se kreću 2D ravninom, konfiguracijski prostor uglavnom se sastoji od triju dimenzije: položaja robota (x, y) i njegove orijentacije θ. Orijentacija θ odnosi se na smjer u kojem je robot okrenut, što je važno za planiranje kretanja zato što robot ne može trenutačno promijeniti smjer bez pomaka, odnosno ima određena ograničenja u kretanju. Prikaz konfiguracijskog prostora za različite orijentacije robota nalazi se na slici 2.5. Slično radnom prostoru, konfiguracijski prostor također se može podijeliti na slobodni prostor (C_{free}) i prostor prepreka (C_{obs}). Slobodan prostor u konfiguracijskom prostoru sastoji se od svih konfiguracija u kojima se robot može kretati bez sudara s preprekama. To odgovara konfiguracijama u kojima se cijelo tijelo robota, na temelju njegove geometrije i orijentacije, nalazi u slobodnom prostoru radnog prostora. Zauzeti prostor u konfiguracijskom prostoru uključuje sve konfiguracije u kojima bi se robot sudario s preprekom. Planiranje putanje kod mobilnih robota može se pojednostaviti predstavljanjem problema u konfiguracijskom prostoru. Algoritmi poput A*, Rapidly-exploring Random Trees (RRT) ili Probabilistic Roadmaps (PRM) obično se koriste za planiranje putanje u C-prostoru.



Slika 2.5. Prikaz konfiguracijskog prostora [6]

2.4. Određivanje konfiguracijskog prostora

Kao što je već navedeno radni prostor robota predstavlja fizičko okruženje u kojem se robot kreće i djeluje. U tipičnom radnom prostoru, prepreke su predstavljene kao regije prostora koje središte ili tijelo robota ne može presjeći. Kako bi se iz radnog prostora dobio konfiguracijski prostor, u ovom radu korištena je tehnika proširivanja prepreka (engl. *Obstacle inflation*). Ključan izazov u stvaranju konfiguracijskog prostora iz radnog prostora jest taj što robot nije točkasti objekt, već ima fizičku veličinu i oblik, što znači da mu je potrebna određena količina prostora da se kreće bez sudara s preprekama. Ovdje dolazi do izražaja proširivanje prepreka. Proširivanje prepreka uključuje povećanje veličine svake prepreke u radnom prostoru za marginu jednaku veličini robota ili sigurnosnoj udaljenosti. Time se pojednostavljuje planiranje puta transformirajući problem u onaj gdje se robot može tretirati kao točka u konfiguracijskom prostoru. Prošireno područje osigurava da robot, kada planira put kroz konfiguracijski prostor, ostane na sigurnoj udaljenosti od prepreka i izbjegava sudare, čak i kada se uzmu u obzir njegove pune dimenzije. Proširivanjem prepreka osigurava se da roboti mogu kretati autonomno i sigurno u složenim i dinamičnim okruženjima.



Slika 2.6. Određivanje konfiguracijskog prostora

Prvi je korak kod proširivanja prepreka odrediti veličinu i oblik robota. Na primjer, mobilni robot može imati kružni, pravokutni ili složeniji oblik. Poznavanje dimenzija robota ključno je jer se proširivanje prepreke mora izvršiti u skladu sa stvarnom veličinom robota. Ako robot ima kružni oblik, proširivanje prepreka poprilično je jednostavno jer je potrebno samo odrediti polumjer robota. Kod robota pravokutnog ili nepravilnog oblika proces proširivanja složeniji je i može uključivati izračun maksimalnog razmaka potrebnog u svim mogućim usmjerenjima. Nakon što su poznati veličina i oblik robota, sljedeći je korak proširiti prepreke u radnom prostoru. Svaka je prepreka napuhana za marginu jednaku veličini robota ili sigurnosnoj udaljenosti. Proces proširivanja prepreka za kružne robote ostvaruje se tako da se svaka prepreka ravnomjerno proširi dodavanjem margine jednake polumjeru robota. Za pravokutne ili nepravilne robote prepreke se moraju širiti u različitim smjerovima kako bi se uzele u obzir moguće orijentacije robota. Proširivanje mora osigurati da u bilo kojoj mogućoj orijentaciji robot neće presjeći prepreku. Napuhane prepreke sada se tretiraju kao nove prepreke u konfiguracijskom prostoru. Primjer proširivanja prepreka za kružnog robota prikazan je na slici 2.6.

3. Planiranje putanje

Temeljni problem svakog robotskog sustava predstavlja proces generiranja putanje koju robot treba slijediti. Problematika planiranja putanje uključuje planiranje putanje od početne do ciljne pozicije, uz izbjegavanje prepreka i poštivanje ograničenja. Planiranjem putanje izračunava se slijed točaka, odnosno pozicija koje robot slijedi. Putanja koja povezuje navedene točke, mora zadovoljiti kinematička i dinamička ograničenja, osigurati da se robot ne sudari s preprekama ili drugim pokretnim objektima u okruženju i optimizaciju prema određenoj metrici.Planiranje putanje može se podijeliti na lokalno i globalno planiranje putanje.

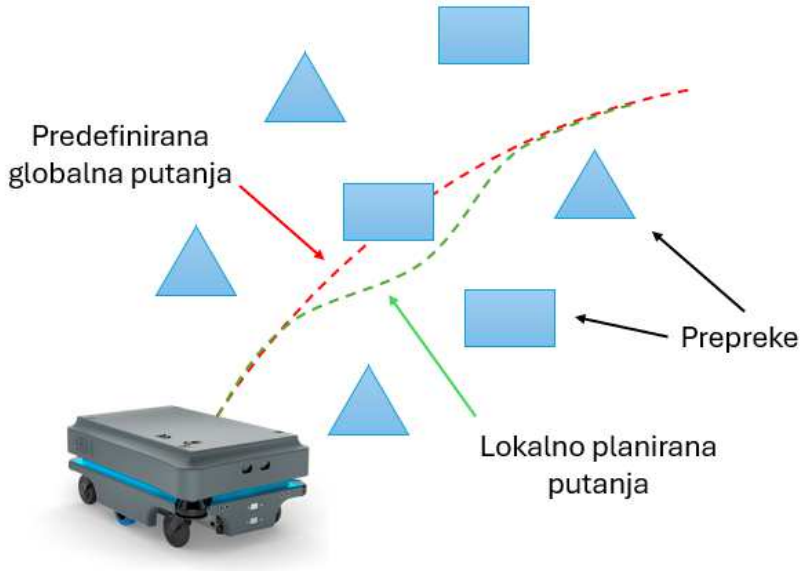
3.1. Lokalno planiranje putanje

Lokalno planiranje putanje bavi se navigacijom u stvarnom vremenu u dinamičnim ili djelomično poznatim okruženjima. Ponajprije se određuje smjer kretanja i brzina na temelju lokalnog okruženja oko mobilnog robota[7]. U ovom slučaju robot koristi podatke sa senzora kako bi percipirao svoju okolinu i, sukladno njoj, donio odluke o tome kako izbjeći prepreke. Lokalnim planiranjem putanje kontinuirano se prilagođava putanja robota na temelju novostečenih informacija. Ovaj je pristup posebno koristan u okruženjima u kojima se prepreke neprestano pomiču, ali ne i u onima s kompleksnim rasporedom prepreka.

3.2. Globalno planiranje putanje

Globalno planiranje putanje vrši se u unaprijed poznatim okruženjima i planirana putanja navodi robota od trenutačne pozicije do cilja.[7] Ova vrsta planiranja pretpostavlja da robot ima potpune informacije o svojoj okolini i da može unaprijed izračunati optimalnu putanju. Uglavnom se koristi u statičnim okruženjima gdje su prepreke statične zbog nemogućnosti određivanja smjera kretanja i brzine.

Usporedba putanje dobivene globalnim i lokalnim planiranjem putanje može se vidjeti na slici 3.1. Globalnim planiranjem putanje unaprijed se izračunala putanja prikazana crvenom bojom koju naknadno prepriječi pomična prepreka. Lokalnim planiranjem putanje navedena prepreka uspješno se izbjegava zbog kontinuiranog prilagođavanja putanje na informacije iz okoline. Time je naglašena važnost korištenja globalnog planiranja putanje u statičnim i unaprijed poznatim okruženjima dok se lokalnom planiranju putanje pridaje prednost u dinamičkim okruženjima.



Slika 3.1. Usporedba lokalnog i globalnog planiranja putanje

3.3. Algoritmi planiranja putanje

Brojni različiti algoritmi razvijeni su za rješavanje problema planiranja putanje od kojih svaki odgovara specifičnim okruženjima i zahtjevima.

3.3.1. Algoritmi lokalnog planiranja putanje

Najčešće korišteni algoritmi za planiranje putanje koji se koriste kod lokalnog planiranja putanje su "*Bug*" algoritmi i Potencijalna polja.

Potencijalna polja

Algoritam potencijalnih polja okruženje robota prikazuje kao polje magnetskih silnica. Prepreke i mobilni robot imaju isti polaritet, a ciljna pozicija suprotan polaritet time stvarajući polje potencijalnih sila privlačenja i odbijanja koje usmjeravaju kretanje robota [8]. Robot predstavlja masu kojom se upravlja rezultirajućim virtualnim poljem sila. Algoritam je jednostavan za implementaciju i ima odziv u stvarnom vremenu, zbog čega je iznimno popularan, ali treba voditi računa o problemu lokalnih minimuma.

"*Bug*" algoritmi

Bug algoritmi jednostavni su diskretni algoritmi korišteni u okruženjima gdje je poznavanje prostora ograničeno. Suprotno okruženju, smjer prema cilju poznat je i sukladno njemu robot započinje svoje kretanje. Robot se kreće sve dok ne naiđe na prepreku, a ovisno o tipu "*Bug*" algoritma koji se koristi, kretanje nastavlja nakon potpunog ili djelomičnog obilaska prepreke.

3.3.2. Algoritmi globalnog planiranja putanje

Globalno planiranje putanje koristi se kod unaprijed poznatih okruženja u kojima se robot kreće. Algoritmi globalnog planiranja putanje traže optimalnu putanju na temelju specifičnih kriterija kao što su najkraća udaljenost, minimalno vrijeme ili energetska učinkovitost. U nastavku je navedena podjela navedenih algoritama.

"Roadmap" alogritmi

"Roadmap" alogritmi koriste se u slučajevima kada je prostor u kojem se robot giba visoko kompleksan i visoko dimenzionalan. Prednost "Roadmap" algoritma jest u tome što ne predstavlja eksplicitno cijeli konfiguracijski prostor, već odabire pojedine točke koje povezuje u graf putanja robota. Time se planiranje putanje svodi na pretraživanje grafa. Najčešće su korišteni primjeri grafova koji se koriste kod takvih algoritama Poopćeni Voronoi graf i Vizibilit graf.

Planiranje putanje uzorkovanjem prostora

Ovakav način planiranja putanje predstavlja potpuno drugačiji pogled od dosadašnjih koji su se ograničavali na prikaz zauzeća prostora. Planiranje putanje uzorkovanjem putanje nasumično uzorkuje konfiguracijski prostor s ciljem povezivanja pojedinih dijelova prostora. Pretpostavka je kako će se nasumičnim odabirom točaka prije otkriti pozicija cilja, ali u slučaju nepostojanja putanje do njega, ovim pristupom ne možemo odrediti njezino postojanje. Algoritmi pronalaska putanja koji koriste ovakav pristup su: Brzorastuća slučajna stabla i Vjerojatnosni "roadmap" algoritmi.

Algoritmi pretraživanja grafova putanja

U robotici prostor se može prikazati kao graf gdje čvorovi označavaju točne pozicije u prostoru, a bridovi između pojedinih čvorova predstavljaju putanje koje ih povezuju. Algoritmi pretraživanja grafova pretražuju bridove i čvorove kako bi pronašli optimalni put od početne do krajnje točke uz izbjegavanje prepreka. Podjela algoritama je u dvije kategorije: informirane i neinformirane. Neinformirani algoritmi istražuju okolinu bez ikakvih dodatnih informacija o točnoj poziciji cilja ili putanjama koje imaju najveću vjerojatnost voditi prema cilju. Ovi algoritmi prikladniji su kada robot ima ograničene informacije o okolini ili mora istraživati potpuno nepoznat prostor. Informirani algoritmi koriste heuristiku za usmjeravanje pretraživanja, što ih čini učinkovitijima. Veći prioritet pridaju putanjama za koje je veća vjerojatnost da će pronaći cilj. Informirani algoritmi korisni su kod izračuna u stvarnom vremenu, gdje su potrebne brze odluke, izbjegavajući nepotrebno istraživanje manje relevantnih područja. Jedan od najčešće korištenih algoritama pretraživanja grafova jest A*. To je informirani algoritam koji koristi heurističku funkciju, kao što je euklidska udaljenost do cilja, kod traženja najbolje putanje.

3.4. Ograničenja planiranja putanje

Planiranje putanje za mobilne robote ne uključuje samo pronalaženje najkraćeg puta. Potrebno je uzeti u obzir i ograničenja kako bi se osiguralo da robot može sigurno i učinkovito slijediti planiranu putanju. Jedna su od najvažnijih ograničenja kinematička i dinamička ograničenja robota. Robot mora biti u mogućnosti slijediti planiranu putanju bez prekoračenja maksimalne brzine, ubrzanja ili ograničenja sile. Neholonomska ograničenja unose kompleksnost u pronalasku rješenja u usporedbi s dosadašnjim ograničenjima, ali su također bitna i moraju biti uzeta u obzir. Osim ograničenja sigurnost je bitna jer robot mora izbjegavati prepreke ne samo održavajući sigurnu udaljenost od njih, već i osiguravajući da se može zaustaviti ili promijeniti smjer ako se nova prepreka iznenada pojavi. To predstavlja izazov u dinamičnim okruženjima gdje se prepreke neprestano kreću. Planiranje putanje u takvim okruženjima zahtijeva kontinuirano ažuriranje prikaza prostora na temelju podataka senzora u stvarnom vremenu. Zbog nesigurnosti senzora prikazi okruženja mogu biti netočni ili zašumljeni, što može uzrokovati pogreške pri izračunu putanje i izbjegavanju prepreka. Prikaz u stvarnom vremenu zahtijeva algoritme koji su visoke računalne složenosti, zbog čega je ključno balansiranje kompromisa između računalne složenosti i optimalnosti.

4. A* algoritam

4.1. O algoritmu

Algoritam A* jedan je od najčešće korištenih i najpoznatijih algoritama za pronalaženje putanje. Spada u skupinu algoritama temeljenih na pretraživanju grafova putanja, detaljnije u skupinu informiranih algoritama pretraživanja grafova putanje. Naziv informirani dolazi od činjenice da pri traženju putanje A* algoritam koristi heurističku funkciju. Heuristička funkcija izračunava koliko je svaka točka u prostoru djelovanja robota udaljena od točke u kojoj se nalazi zadana ciljna pozicija. Pomoću heurističke funkcije određuje se najbolji smjer traženja sljedeće točke putanje[9]. Heuristička funkcija dio je evaluacijske funkcije koja za svaki čvor u prostoru procjenjuje iznos cijene putanje koja u sebi sadržava promatrani čvor[10]. Evaluacijska funkcija definirana je na sljedeći način:

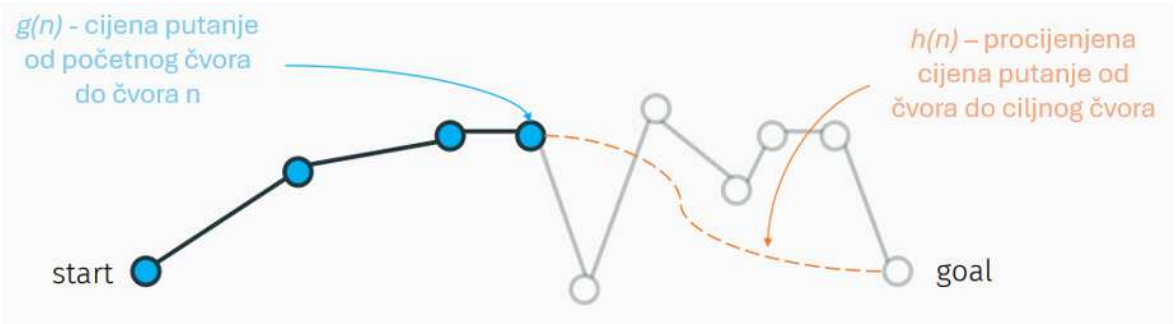
$$f(n) = h(n) + g(n) \tag{4.1}$$

gdje je $f(n)$ vrijednost evaluacijske funkcije promatranog čvora n; $h(n)$ je procijenjena vrijednost cijene od n do ciljnog čvora; $g(n)$ vrijednost cijene od početnog čvora do n, kako je prikazano na slici 4.1. Procjena heurističke funkcije računa se euklidskom, manhattanskom ili chebyshevom mjerom udaljenosti između dvaju čvora. U ovom radu korištena je euklidska mjera udaljenosti:

$$h(n) = \sqrt{(x_n - x_{cilj})^2 + (y_n - y_{cilj})^2} \tag{4.2}$$

Heuristička funkcija mora zadovoljavati svojstva dopustivosti i konzistentnosti kako bi algoritam pretraživao najmanji broj čvorova pri pronalasku optimalne putanje. Do-

pustivost osigurava da se ne precjenjuje optimalna putanja od čvora n do ciljnog čvora. Ako je heuristička funkcija dopustiva tada je algoritam A* dopustiv čime se osigurava pronalazak optimalne putanje. Konzistentnost nalaže da svaka procjena heurističke funkcije ne smije biti veća od zbroja procjene heurističke funkcije za susjedni čvor i cijene puta od trenutnog do susjednog čvora[10]. Ispravan odabir heurističke funkcije, odnosno evaluacijske funkcije, utječe na uspješnost i vrijeme potrebno pri pronalasku optimalne putanje.



Slika 4.1. Vizualizacija komponenti evaluacijske funkcije [11]

Implementacija A* algoritma prikazana pseudokodom 1 temelji se na otvorenoj (engl. *Open*) i zatvorenoj (engl. *Closed*) listi. Otvorena lista sadrži sve čvorove koji su susjedni čvorovi dosad posječenih čvorova, ali kojima još nije pristupljeno. Čvorovi otvorene liste poredani su s obzirom na vrijednost evaluacijske funkcije od najmanje do najveće vrijednosti. Zatvorena lista sadrži sve čvorove kojima je već pristupljeno.

Kod A* algoritma prvo se zadaju početni i ciljni čvor na temelju pozicije u prostoru od koje do koje se robot giba. Početni čvor dodaje se u otvorenu listu dok zatvorena lista ostaje prazna. Vrijednost evaluacijske funkcije početnog čvora jednaka je vrijednosti heurističke funkcije zato što je cijena početnog čvora jednaka nuli. Dok otvorena lista nije prazna uzima se prvi čvor s otvorene liste i dodaje u zatvorenu listu. U slučaju kada je navedeni čvor ciljni čvor, uspješno je izračunata najoptimalnija putanja. U suprotnom se uzima svaki slobodni susjedni čvor trenutnog čvora i redom se provjeravaju uvjeti. U slučaju kada je susjedni čvor već sadržan u zatvorenoj listi prelazi se na sljedeći susjedni čvor. Ako je čvor već sadržan u otvorenoj listi, ali mu je vrijednost funkcije $g(m)$ manje od postojeće, tada mu se sprema novo dobivena vrijednost. U slučaju kada čvor ne pripada ni jednoj listi računa se vrijednost evaluacijske funkcije i njezinih sastavnica te

se čvor dodaje u otvorenu listu. Na temelju izračunate vrijednosti evaluacijske funkcije odabire se sljedeći čvor na koji se proširuje putanja tako što se odabire čvor s najmanjom vrijednošću funkcije.

Pseudokod 1 A* algoritam

Ulaz: Početni čvor $start$, ciljni čvor $goal$
Izlaz: Najoptimalniji put od $start$ do $goal$

Otvoreni skup $\mathcal{O} = \{start\}$
Zatvoreni skup $\mathcal{C} = \emptyset$
$g(start) = 0$ ▷ Vrijednost cijene početnog čvora
$f(start) = g(start) + h(start)$ ▷ Evaluacijska funkcija f
while $\mathcal{O}$ nije prazan **do**
    Odaberi čvor n iz $\mathcal{O}$ sa najmanjom vrijednošću $f(n)$
    Ukloni n iz $\mathcal{O}$ i dodaj u $\mathcal{C}$
    if $n = goal$ **then**
        return putanja ▷ Ako smo došli do cilja, vrati putanju
    end if
    for svaki susjed m čvora n **do**
        if $m \in \mathcal{C}$ **then**
            continue
        end if
        $g_{novi} = g(n) + cost(n, m)$
        if $m \notin \mathcal{O}$ **and** $m \notin \mathcal{C}$ **then**
            Dodaj m u $\mathcal{O}$
            $g(m) = g_{novi}$
            $h(m) = heuristika(m, goal)$
            $f(m) = g(m) + h(m)$
        end if
        if $m \in \mathcal{O}$ **and** $g_{novi} < g(m)$ **then**
            Spremi novu vrijednost funkcije f za m
        end if
    end for
end while
vrati neuspjeh ▷ Ako nije pronađen put

4.2. Prednosti i nedostatci

Jedna od primarnih prednosti A* algoritma jest njegova optimalnost. A* osigurava pronalazak optimalne putanje, pod uvjetom da heuristička funkcija nikada ne precjenjuje stvarnu cijenu postizanja cilja. Zbog toga u okruženjima u kojima je cijena kretanja promjenjiva, A* može pronaći najbolje rješenje, osiguravajući da robot uvijek slijedi optimalnu rutu. Još jedna značajna prednost jest činjenica da ako rješenje postoji, algoritam će ga uvijek pronaći. Učinkovitost A* algoritma poboljšana je korištenjem heurističke funkcije jer smanjuje broj nepotrebnih čvorova koji se istražuju. To čini algoritam učinkovitijim u usporedbi s neinformiranim metodama pretraživanja. Heuristička funkcija može se prilagoditi specifičnom problemu, čineći A* fleksibilnim. Heuristička funkcija i funkcije koje su sadržane unutar evaluacijske funkcije,a koje se koriste kao kriterij, mogu se usvojiti, modificirati ili dodati druge funkcije. Ovo daje širok raspon modifikacija osnovnog principa jer se vrijeme, potrošnja energije ili sigurnost također mogu uključiti u funkciju f(v).[12]

Usprkos navedenom, A* ima znatna ograničenja, posebice u pogledu upotrebe memorije. Budući da algoritam pohranjuje sve čvorove koje istražuje u otvorenoj i zatvorenoj listi, može postati iscrpan za memoriju, posebno u velikim ili složenim okruženjima. To može predstavljati veliki problem u sustavima s ograničenim resursima, gdje je dostupnost memorije ograničena. Iako je učinkovitiji od neinformiranih metoda, njegova izvedba dugotrajna je u primjeni u velikim prostorima s brojnim čvorovima i vezama. To je najveći izazov kod primjene u stvarnom vremenu gdje je brzo donošenje odluka ključno. Osim toga, A* je manje prikladan i za dinamična okruženja. U dinamičnim okruženjima, gdje se prepreke ili drugi čimbenici mogu promijeniti tijekom kretanja robota, algoritam se mora često ponovno pokretati kako bi se prilagodio tim promjenama. To može dovesti do neučinkovitosti jer ponovno izračunavanje putanja u stvarnom vremenu dodaje značajne zahtjeve za memoriju.

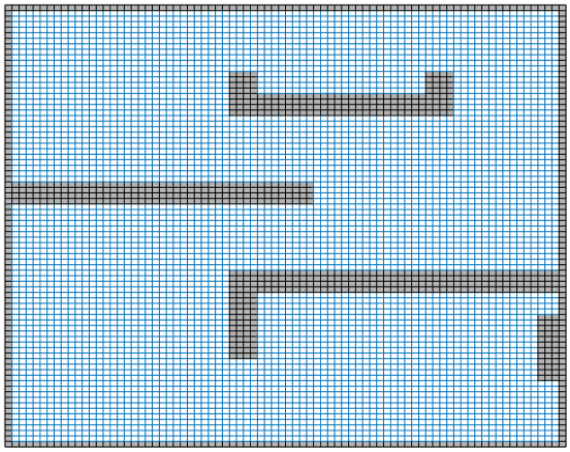
A* uspostavlja ravnotežu između iscrpnog istraživanja i heurističkog usmjeravanja. Njegove prednosti leže u njegovoj sposobnosti pronalaženja optimalnih putova i fleksibilnosti u prilagodbi različitim okruženjima putem heurističkih funkcija. Međutim, izvedba algoritma ovisi o korištenju memorije, računalnoj učinkovitosti i kvaliteti heurističke funkcije.

5. Prilagodba A* algoritma

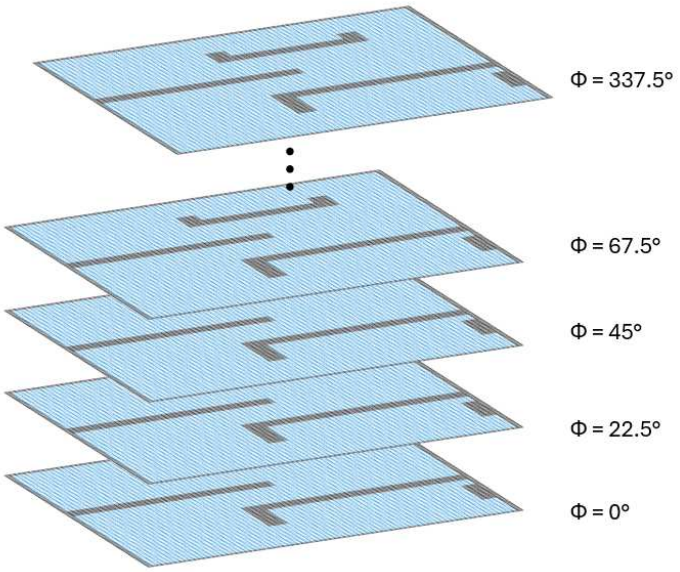
Prethodno navedeni A* algoritam pretraživanja grafova putanja kod planiranja putanje ne uzima u obzir orijentaciju robota. U prostorima ograničene orijentacije, odnosno kada robot sadrži teret nepravilnog oblika potrebno je u izračun putanje uključiti i orijentaciju kako bi se potrebna putanja uspješno izračunala bez sudara s preprekama. Prvi korak prilagodbe algoritma, kako bi u obzir uzeo i orijentaciju, jest izgradnja konfiguracijskog prostora sukladno orijentacijskim zahtjevima. U ovom radu koristi se A* algoritam implementiran u programu Matlab.

5.1. Izgradnja karte konfiguracijskog prostora

Diskretna metrička karta prostora prikazana je kao matrica gdje vrijednost 0 predstavlja slobodni prostor, a vrijednost 1 predstavlja zauzeti prostor. Izgradnjom konfiguracijskog prostora prepreke u diskretno kontinuiranim metričkim kartama prostora proširuju se za iznos polumjera mobilnog robota. U ovom radu prvo je neproširena matrica kopirana u 16 slojeva čime se dobiva matrica dimenzija 16 x širina x duljina, gdje širina i duljina predstavljaju dimenzije radnog prostora, odnosno dimenzije diskretne metričke karte. Dobivena matrica vizualizirana je na slici 5.1. Svih 16 slojeva matrice vertikalno naslaganih pretvara konfiguracijski prostor iz dvodimenzionalnog u trodimenzionalni kao što je to napravljeno i u radu [13]. 16 slojeva predstavlja 16 orijentacija u kojima se robota može pozicionirati. Navedena podjela od 16 slojeva proizvoljno je odabrana i po potrebi može biti veća ili manja, ali za potrebe ovog rada po uzoru na rad [6] koristimo 16 slojeva. Razlika između dvaju susjednih slojeva predstavlja razliku u orijentaciji robota od 22.5°. Nakon stvaranja potrebnih slojeva za svaki od njih potrebno je proširiti prepreke za zadani otisak robota kako bi se dobio konfiguracijski prikaz prostora koji će se dalje koristiti za traženje putanja.



(a) 2D matrica zauzeća prostora



(b) 16 slojeva matrice zauzeća prostora

Slika 5.1. Prikaz karte zauzeća prostora

5.1.1. Otisak mobilnog robota

Otisci robota korišteni za daljnje izračune zadani su u obliku pravokutnika i slova 'T'. Sukladno potrebama navedeni otisak može se zadati i drugačijeg oblika dok god se može definirati pomoću duljina stranica, iz kojih zatim definiramo slijed točaka koje spojene predstavljaju obrub tereta, ili pomoću kontinuiranih funkcija. Otisci korišteni u ovom radu zadani su na sljedeći način:

Pravokutni otisak

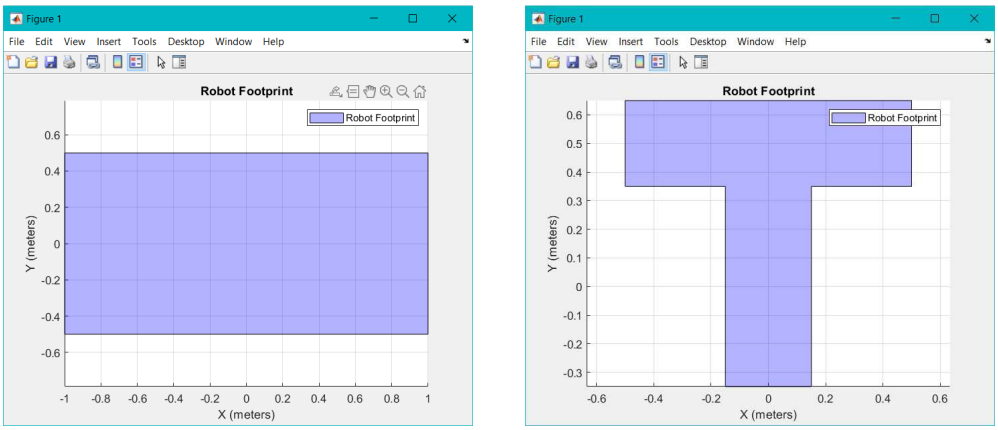
Navedeni otisak zadan je pomoću duljina stranica kojima se definira duljina i širina stranica pravokutnika. Duljine stranica dijele se na pola i dobivene polovice predstavljaju udaljenost od x i y osi do vrhova pravokutnika čiji se centroid nalazi u točci (0,0). Prikaz navedenog pravokutnika može se vidjeti na slici 5.2.a, a odsječak korištenog algoritma na prikazu 5..1 Točke koje predstavljaju vrhove pravokutnika spremljene su u matrici na sljedeći način:

Algoritam 5..1: Isječak algoritma koji prikazuje stvaranje tereta pravokutnog oblika

```
dimensions=[1,0.5];
robotx = dimensions(1);
roboty = dimensions(2);
footprint = [-robotx/2, -roboty/2;
              robotx/2, -roboty/2;
              robotx/2,  roboty/2;
             -robotx/2,  roboty/2];
```

'T' otisak

Kako bi se prikazao otisak u obliku slova 'T', kao i kod pravokutnog oblika prvo se zadaju duljine i širine. Definirane duljine i širine predstavljaju duljinu i širinu vertikalnog i horizontalnog dijela slova 'T', čime se oblik tereta zapravo dijeli na dva pravokutnika. Iz definiranih duljina i širina ponovno dobivamo slijed točaka koji predstavlja obrub tereta čime je postupak definiranja sličan algoritmu 5..1, ali matrica se u ovom slučaju sastoji od osam redaka, za razliku od prethodnog slučaja gdje su samo četiri. Dobiveni otisak prikazan je na slici 5.2.b



(a) Pravokutni otisak **(b)** Otisak u obliku slova 'T'

Slika 5.2. Prikaz korištenih otisaka

5.1.2. Proširivanje prepreka

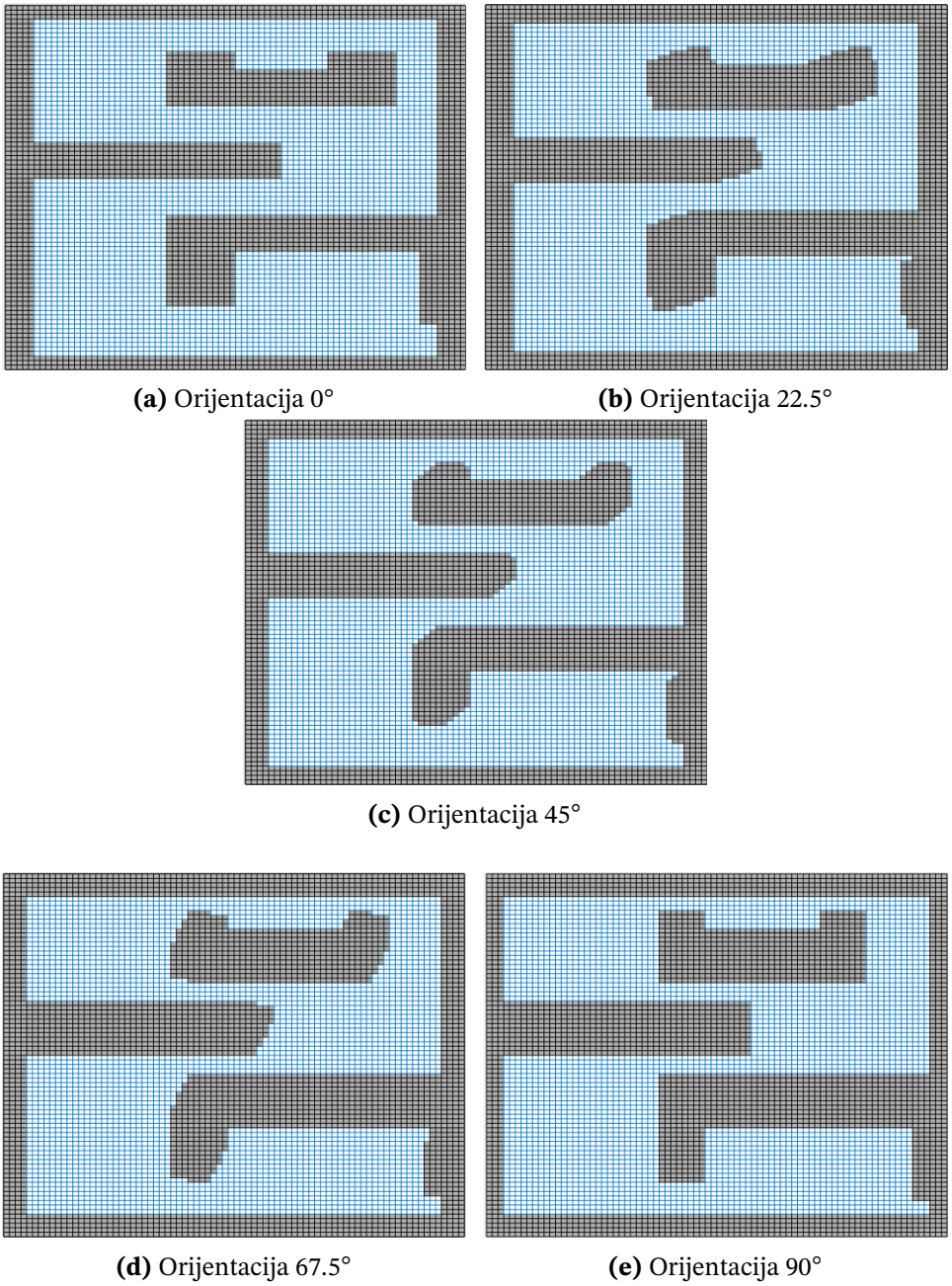
Nakon uspješne definicije, otisak robota s teretom zadan pomoću matrice prosljeđuje se kao argument funkciji *environment* A* star algoritma. *Environment* funkcija gradi konfiguracijski prostor na temelju diskretne metričke karte s definiranim preprekama i otiska robota, a njezin isječak prikazan je algoritmom 5..2 Kao što je prethodno navedeno kreira se matrica od 16 slojeva kako je prikazano u retku 12 algoritma 5..2 Za svaki orijentacijski sloj od jedan do šesnaest definira se rotacijska matrica ovisno o orijentaciji koja se množi s inverzom matrice otiska i zatim se za dobivenu matricu računa inverz koji predstavlja rotirani otisak robota s teretom. Rotirani otisak translatira se u središte svake ćelije čija je vrijednost jedan, odnosno u središte ćelije koja je zauzeta. Kod otiska u obliku slova 'T' prvo je potrebno izračunati centorid otiska koji se računa pomoću funkcije *centroid(polyshape(footprint))*. Kada se otisak robota s teretom pozicionira u središte navedene ćelije, prolazi se kroz svaku ćeliju karte prostora i središta ćelija koja se nalaze unutar definiranog otiska definiraju se s vrijednošću jedan. Pripada li pojedina ćelija unutar ili izvan otiska, provjerava se pomoću funkcije *inpolygon* definirane u *Matlabu*. Time se proširuje prostor prepreke za otisak robota i kreira karta konfiguracijskog prostora za svaku pojedinu orijentaciju.

Algoritam 5..2: Isječak algoritma koji prikazuje kreiranje konfiguracijske karte prostora

```
function environment(obj,limits,res,Rect,footprint)
       obj.x_min = limits(1);obj.x_max = limits(2);obj.y_min = limits(3); obj.y_max = limits(4);
       obj.resolution = res; % res=100 -> 100 cells per meter
       x_delta = obj.x_max - obj.x_min; y_delta = obj.y_max - obj.y_min;
       obj.free = 0; obj.obst = 1;
       obj.map = ones(ceil(y_delta*res),ceil(x_delta*res))*obj.free;
       obj.map(1,:)=obj.obst; obj.map(end,:)=obj.obst; obj.map(:,1)=obj.obst; obj.map(:,end)=obj.obst;
       for i=1:size(Rect,1)
            obj.map( ceil(Rect(i,1)*res+1):ceil(Rect(i,2)*res),...
                    ceil(Rect(i,3)*res+1):ceil(Rect(i,4)*res) ) = obj.obst;
       end
       obj.map = repmat(obj.map, 1, 1, 16); %creating 3D matrix with 16 layers
       Nx=size(obj.map,1);
       Ny=size(obj.map,2);
       inflationmap=obj.map;
       for k=1:length(obj.orientationSteps)
           theta = obj.orientationSteps(mod(k+8-1,16)+1);
           R = [cos(theta), -sin(theta); % Rotation matrix for the given orientation
                sin(theta), cos(theta)];
           rotated_footprint = (R * footprint')'; % Rotate the footprint based on the orientation
           for i=1:Nx
               for j=1:Ny
                   if (obj.map(i,j,k)) == 1
                       dd= 1/obj.resolution/2; %increment
                       gx= 1/obj.resolution*i-dd;
                       gy= 1/obj.resolution*j-dd;
                       translated_footprint(:, 1) = rotated_footprint(:, 1) + gx;
                       translated_footprint(:, 2) = rotated_footprint(:, 2) + gy;
                       if i/obj.resolution>obj.x_min && i/obj.resolution<=obj.x_max &&
                            j/obj.resolution>obj.y_min && j/obj.resolution<=obj.y_max
                           for l=1:Nx
                               for m=1:Ny
                                   hx= 1/obj.resolution*l-dd;
                                   hy= 1/obj.resolution*m-dd;
                                   if inpolygon(hx, hy, translated_footprint(:,1),
                                        translated_footprint(:,2))
                                       inflationmap(l,m,k)=1;
                                   end
                               end
                           end
                       end
                   end
               end
           end
       end
       obj.map=inflationmap;
   end
```

5.1.3. Karta konfiguracijskog prostora

Karta konfiguracijskog prostora drugačija je za svih 16 slojeva karte zbog različite orijentacije u svakom od slojeva. Također, ovisno o otisku robota karta se mijenja pa tako na slikama 5.3. i 5.4. vidi se kako ovisno o otisku u obliku pravokutnika ili slova 'T' karta konfiguracijskog prostora mijenja se za prvih 5 orijentacija robota. Prva orijentacija predstavlja orijentaciju pod kutom od 0° koja predstavlja robota s orijentacijom prema smjeru istoka. Ostale orijentacije redom predstavljaju robota s orijentacijom od 22.5°, 45°, 67.5°, 90°.

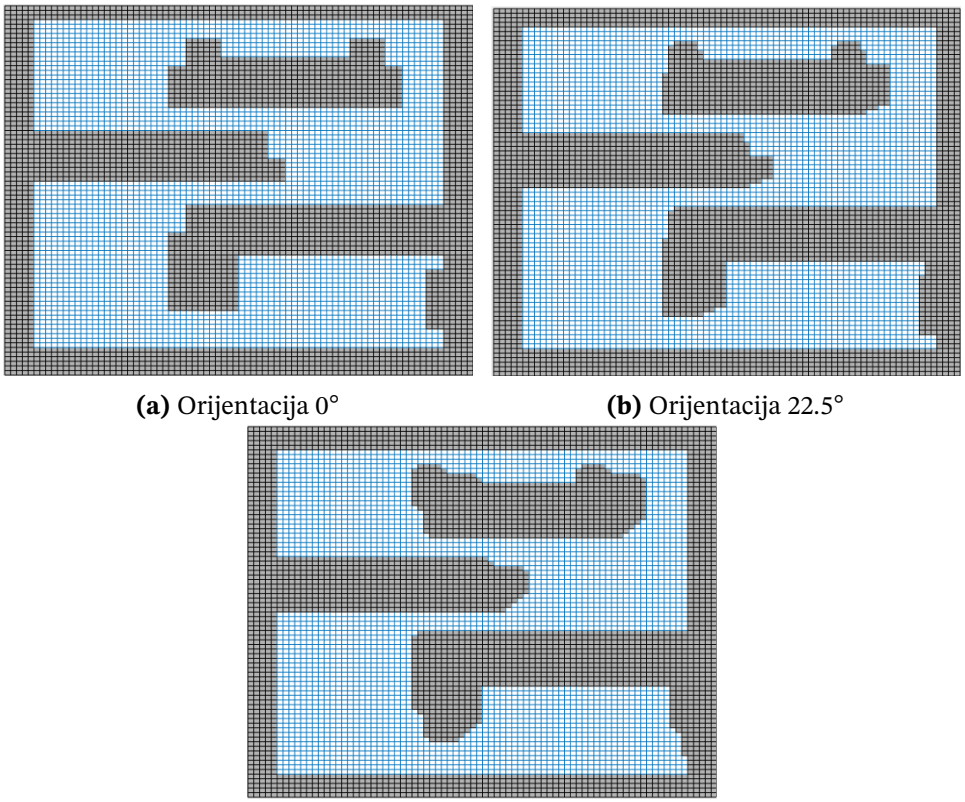


(a) Orijentacija 0° **(b)** Orijentacija 22.5°

(c) Orijentacija 45°

(d) Orijentacija 67.5° **(e)** Orijentacija 90°

Slika 5.3. Karte prvih 5 slojeva konfiguracijskog prostora za robot s teretom pravokutnog oblika



(a) Orijentacija 0° **(b)** Orijentacija 22.5°

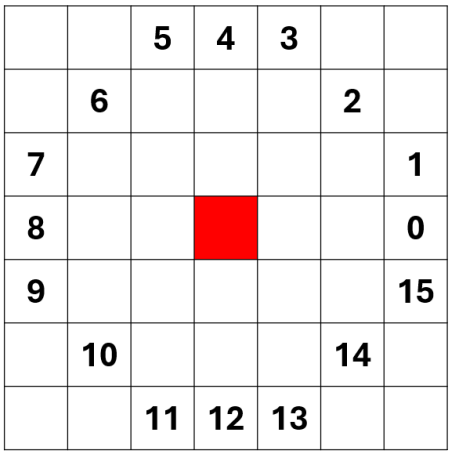
(c) Orijentacija 45°

(d) Orijentacija 67.5° **(e)** Orijentacija 90°

Slika 5.4. Karte prvih 5 slojeva konfiguracijskog prostora za robot s teretom oblika slova ’T’

5.2. Traženje putanje

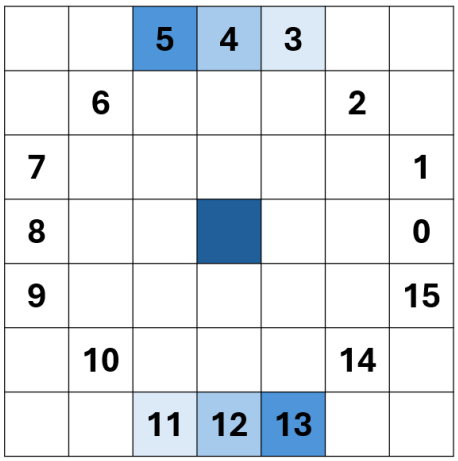
Traženje putanje vrši se pomoću funkcije *find* A* algoritma koja kao parametre prima početnu i ciljnu poziciju, odnosno točku na karti, i početnu i ciljnu orijentaciju robota koja se proizvoljno zadaje. A* algoritam opisan u poglavlju 4. uglavnom pretražuje 4 ili 8 susjednih čvorova s obzirom na trenutni čvor. U ovom radu kod je prilagođen po uzoru na [6] tako da algoritam pretražuje šest susjednih čvorova iz 16 slojeva konfiguracijske karte kreirane u prethodnom koraku. 16 susjednih čvorova i njihov raspored prikazani su na slici 5.5.



Slika 5.5. Susjedni čvorovi promatranog čvora

S obzirom na trenutnu orijentaciju i čvor u kojem se robot nalazi pretražuju se dva čvora u trenutnom sloju orijentacije n, dva čvora prethodnog sloja orijentacije n-1 i dva čvora predstojećeg sloja orijentacije n+1. Slojevi orijentacije n-1 i n+1 predstavljaju slojeve čija orijentacija iznosi $\pm$ 22.5° od trenutne orijentacije robota. U svakom se sloju pretražuju dva čvora koja predstavljaju kretanje unaprijed i unatrag, kako je prikazano na slici 5.6. Trenutni čvor vrijednosti orijentacije četiri prema slici rasporeda susjednih čvorova na open listu dodaje parove čvorova 4 i 12, 3 i 11 i 5 i 13. Kako bi se dobile ispravne vrijednosti kod pronalaska čvorova koji se pretražuju, koristimo *Matlab* funkciju *mod* koja računa modul vrijednosti n-1 i n+1 jer orijentacija može imati vrijednosti samo između 0 i 15. Količina susjednih čvorova koja se pretražuje, može biti i vrijednost veća od šest, ali navedena vrijednost odabrana je kako bi kretanje bilo kontinuirano bez naglih promjena orijentacija što uzrokuje neprirodno kretanje robota. Isječak algoritma koji prikazuje proces pretraživanja susjednih čvorova prikazan je algoritmom 5..3 Čvo-

rovi koji se dodaju u open listu uspoređuju se primarno na temelju evaluacijske funkcije, a zatim vrijednosti razlike trenutne orijentacije i ciljne orijentacije.



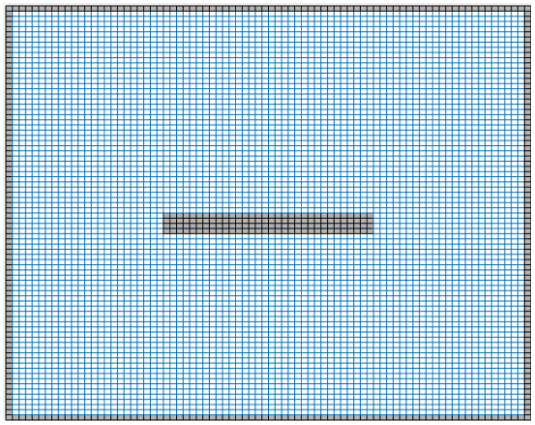
Slika 5.6. Parovi čvorova koji se dodaju na open listu

Algoritam 5..3: Isječak algoritma koji prikazuje iteraciju kroz susjedne čvorove

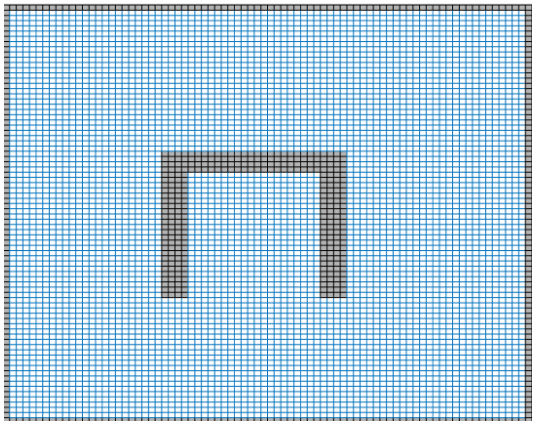
```
for i=-1:1:1
    dxo = dx*[obj.orientationValues(mod(obj.act.orientation-1+i, 16)+1, 1);
         obj.orientationValues(mod(obj.act.orientation-1+i+8, 16)+1, 1)];
    dyo = dy*[obj.orientationValues(mod(obj.act.orientation-1+i, 16)+1, 2);
         obj.orientationValues(mod(obj.act.orientation-1+i+8, 16)+1, 2)];
    nxx=obj.act.pose(1)+dxo; nyy=obj.act.pose(2)+dyo;

    for j=1:length(nxx) % add neighbours
        if j==1
            if obj.addNodeToOpenListG(nxx(j), nyy(j), mod(obj.act.orientation-1+i, 16)+1)
                obj.updateDraw([nxx(j);nyy(j)],[obj.act.pose],2); % draw the last node (new added node)
                % Sort open list
                [~,k] = sortrows([[obj.open.cth]*1+[obj.open.ctg]*1; obj.open.orientation_diff].',
                    [1,2]);
                obj.open = obj.open(k);
            end
        elseif j==2
            if obj.addNodeToOpenListG(nxx(j), nyy(j), mod(obj.act.orientation-1+i+8, 16)+1)
                obj.updateDraw([nxx(j);nyy(j)],[obj.act.pose],2); % draw the last node (new added node)
                % Sort open list
                [~,k] = sortrows([[obj.open.cth]*1+[obj.open.ctg]*1; obj.open.orientation_diff].',
                    [1,2]);
                obj.open = obj.open(k);
            end
        end
    end
end
```

6. Rezultati i rasprava

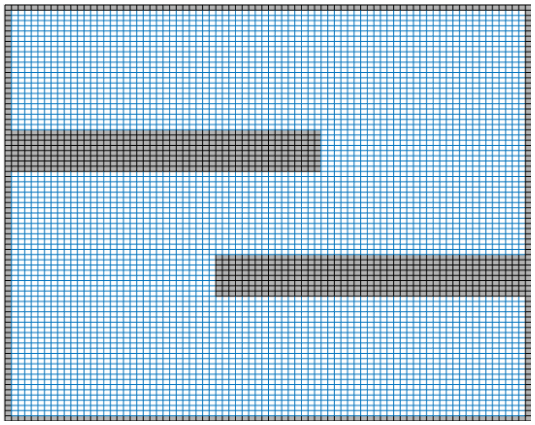
Za vrijeme trajanja izgradnje konfiguracijskog prostora i izračuna konačne putanje od početne do ciljne pozicije mjeri se vrijeme potrebno za izvršavanje svake od tih funkcija. Sva vremena prikazana u nastavku izražena su u sekundama. Nakon svih potrebnih izračuna i pronalaska putanje pomoću funkcije *find* A* algoritma, iscrtava se konačno dobivena putanja mobilnog robota s teretom funkcijom *drawFinalPath*. Svi izračuni vrše se za konstantno jednaku vrijednost rezolucije prostora koja iznosi osam, što predstavlja broj ćelija čija ukupna duljina iznosi jedan metar. Oblici prepreka korišteni u ovom radu prikazani su na slikama6.1., gdje je pomoću karte zauzeća prikazana binarna diskretna metrička reprezentacija prostora. Sivom bojom predstavljen je zauzeti prostor, odnosno prepreke. Slika 6.1.a prikazuje simetričnu jednostavnu prepreku, zatim na sljedećoj slici 6.1.b prepreka je u obliku slova 'U', a dalje redom slijede dva prostora sa složenim preprekama na slikama 6.1.c i 6.1.d i na kraju na slici 6.1.e prepreka je koja stvara uski prolaz. Oblici tereta korišteni za izračun putanje u navedenim prostorima su pravokutnik i oblik slova 'T' čiji su prikazi navedeni u 5.1.1.



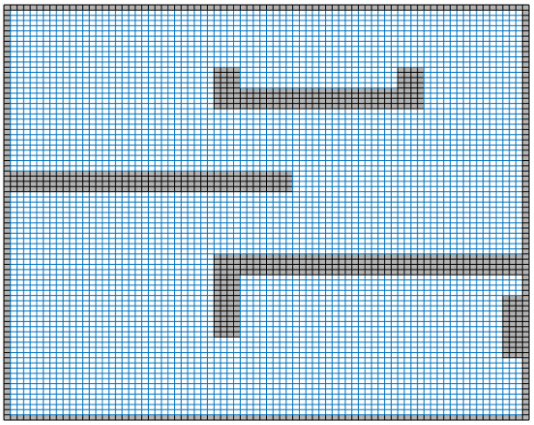
(a) Jednostavna prepreka



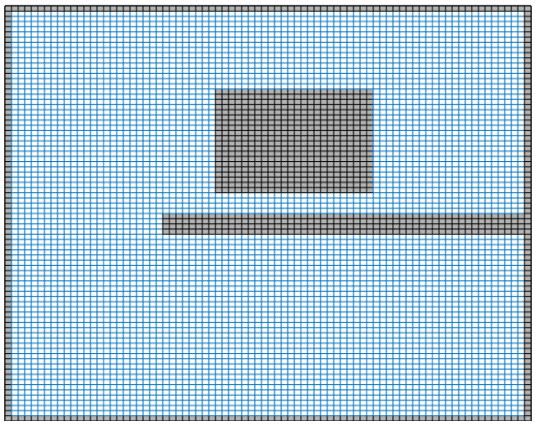
(b) Prepreka oblika slova 'U'



(c) Složene prepreke 1



(d) Složene prepreke 2



(e) Prepreke s uskim prolazom

Slika 6.1. Prikaz karata zauzeća s različitim preprekama

6.1. Izgradnja karte konfiguracijskog prostora

Vrijeme potrebno za izgradnju konfiguracijske karte prostora proporcionalno ovisi o rezoluciji prikaza prostora pa je tako s manjom rezolucijom vrijeme potrebno za izgradnju konfiguracijske karte prostora kraće, a s većom rezolucijom duže. U ovom radu iznos rezolucije konstantno je jednak. Time se osigurava jednaka kompleksnost prikaza prostora i jednak iznos zauzeća memorije kod različitih oblika prepreka. Uz navedeno, na taj je način i omogućena usporedba izgradnje karata vodeći računa o drugim parametrima. Ovisno o obliku prepreka i tereta uspoređuje se navedena vrijednost vremena potrebna za izgradnju konfiguracijske karte prostora. Rezultati svih mjerenja prikazani su u tablici 6.1. Iz navedene tablice vidljivo je kako je za pravokutni oblik tereta, koji je pravilan, vrijeme izgradnje konfiguracijske karte prostora u svim slučajevima kraće nego što je to slučaj s teretom oblika slova 'T'. Najveće odstupanje između dvaju oblika nastupa za prepreke složenog oblika verzije dva. Kod tereta oblika slova 'T' potrebno je više nego dvostruko više vremena za izgradnju karte konfiguracijskog prostora nego što je to slučaj za pravokutni oblik. Kod pravokutnog oblika tereta izgradnja konfiguracijskog prostora vremenski je kraća zbog simetričnosti njegovog oblika. Za vrijednosti orijentacije od 0° do 180° i od 180° do 360° kreiraju se jednake karte konfiguracijskog prostora. Zbog toga je potrebno izgraditi samo one karte konfiguracijskog prostora za vrijednosti orijentacije 0° do 180° i kopirati ih.

Oblik prepreka / Oblik tereta	Jednostavna	'U'	Složene 1	Složene 2	Uski prolaz
Pravokutnik	103.47	92.02	540.27	107.28	124.49
'T'	123.11	104.29	631.55	238.16	218.14

Tablica 6.1. Vrijeme izgradnje karte ovisno o obliku prepreka i tereta

Vrijeme koje je potrebno za izgradnju karte konfiguracijskog prostora može se smanjiti korištenjem računalnog sustava boljih performansi. Usprkos duljem trajanju izračuna, kod implementacije to ne predstavlja ograničenje. Razlog je tomu što jednom izgrađene karte konfiguracijskog prostora nije potrebno ponovno graditi sve dok ne dođe do promjene u zauzeću prostora ili obliku tereta.

6.2. Pronalazak putanje

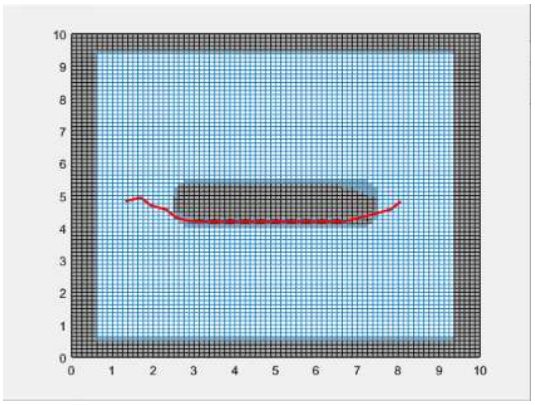
Za svaki od prethodno navedenih oblika prepreka i tereta računa se i vrijeme potrebno za pronalazak putanje od početne do ciljne pozicije. Dobiveno vrijeme uspoređuje se s vremenom potrebnim za pronalazak putanje korištenjem klasičnog A* algoritma koji ne uzima u obzir orijentaciju robota. U tablici 6.2. prikazana su vremena traženja pojedinih putanja pomoću prilagođenog A* algoritma, a na slikama 6.2., 6.3. i 6.4. vizualizacija pronađenih putanja u konfiguracijskom prostoru. Usporedbom dobivenih vremena u tablici 6.2. s vremenima u tablici 6.3. zaključuje se kako vrijeme traženja putanje, za sve oblike prepreka i tereta, pomoću prilagođenog A* algoritma iznosi dulje nego kod klasičnog A* algoritma. Pretpostavka je kako zbog manjeg sveukupnog broja čvorova koji se pretražuju, vrijeme pretrage putanje kod klasičnog A* algoritma iznosi manje nego u odnosu na prilagođeni A* algoritam. Konfiguracijski prostor klasičnog A* algoritma nastaje proširivanjem prepreka za iznos polumjera koji je jednak polumjeru opisane kružnice pojedinog tereta. Opisana kružnica i vrijednost njezinog polumjera izračunate su pomoću funkcije navedene u privitku A . Proširivanje prepreka dobivenim polumjerom rezultira drastičnim smanjenjem slobodnog prostora zbog nepravilnog oblika tereta koji uzrokuje veliki polumjer, a navedena pojava vidljiva je na slici 6.4. Proširivanjem prepreka samo na temelju polumjera nepotrebno se smanjuje radni prostor robota. Kod navedene pojave u slučaju složene prepreke verzije dva prikazane na slici 6.4.d klasični A* algoritam je u nemogućnosti pronaći putanju zbog nepostojanja slobodnog prostora za kretanje robota od početne do ciljne pozicije. Također u slučaju prikazanom na slici 6.4.e uski prolaz postaje u potpunosti zauzet zbog čega dobivena putanja obilazi sustav prepreka.

Oblik prepreka / Oblik tereta	Jednostavna	'U'	Složene 1	Složene 2	Uski prolaz
Pravokutnik	15.95	13.38	10.27	9.82	13.88
'T'	19.43	16.98	12.01	14.35	17.81

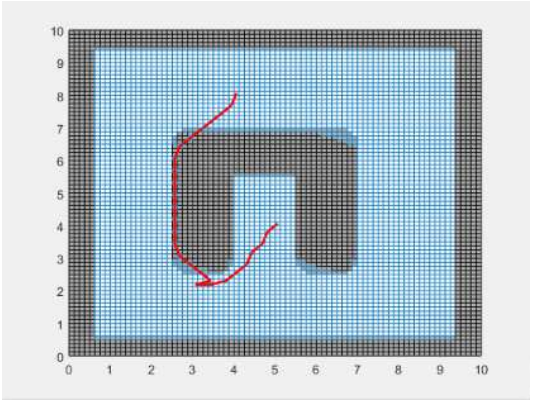
Tablica 6.2. Vrijeme pronalaska putanje ovisno o obliku prepreka i tereta za prilagođeni A* algoritam

Oblik prepreka	Jednostavna	'U'	Složene 1	Složene 2	Uski prolaz
	7.94	4.21	2.95	/	5.17

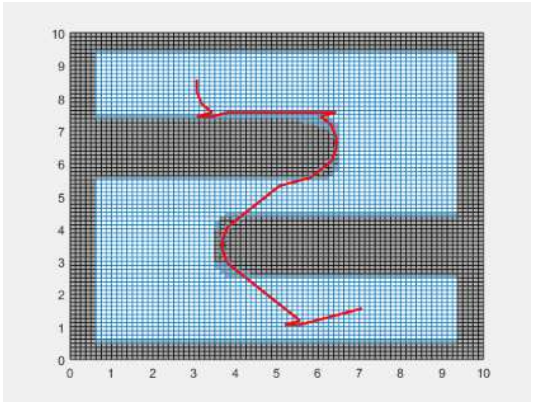
Tablica 6.3. Vrijeme pronalaska putanje ovisno o obliku prepreka za A* algoritam



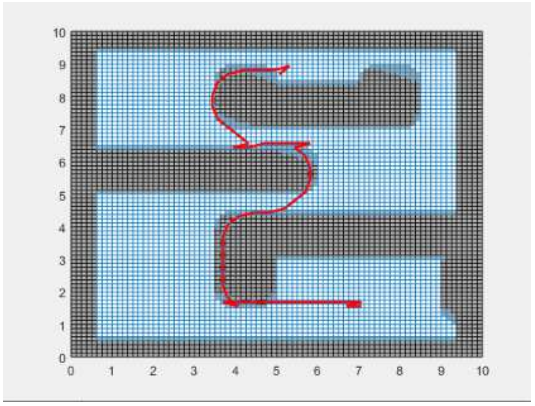
(a) Jednostavna prepreka



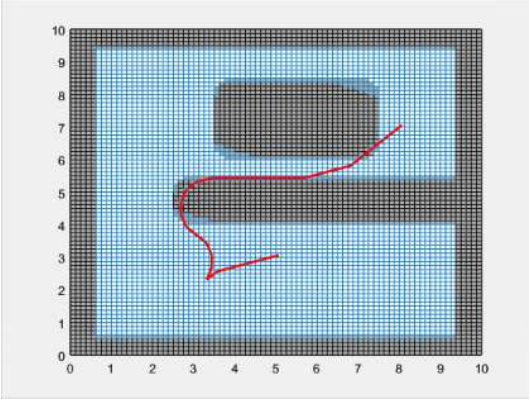
(b) Prepreka oblika slova 'U'



(c) Složene prepreke 1

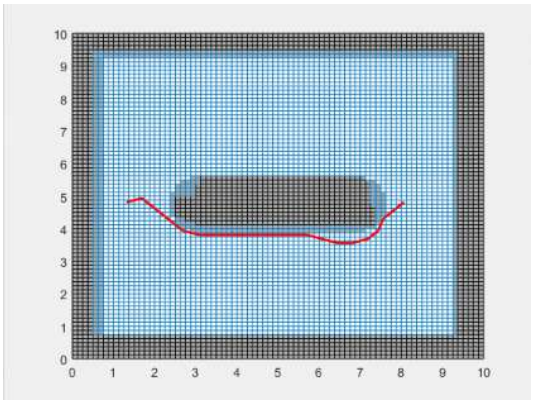


(d) Složene prepreke 2

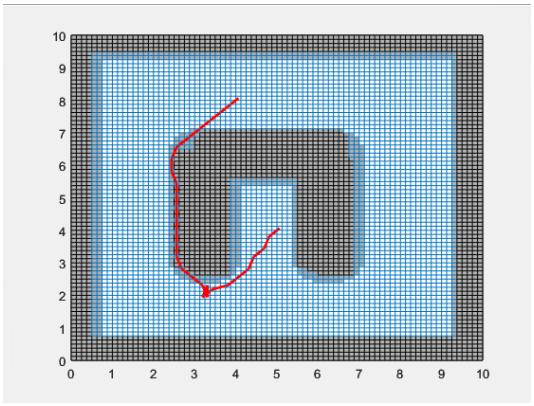


(e) Prepreke s uskim prolazom

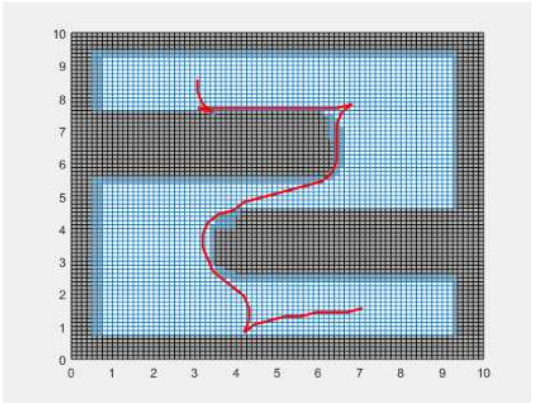
Slika 6.2. Prikaz putanje pronađenih prilagođenim A* algoritmom s različitim preprekama i teretom oblika pravokutnika



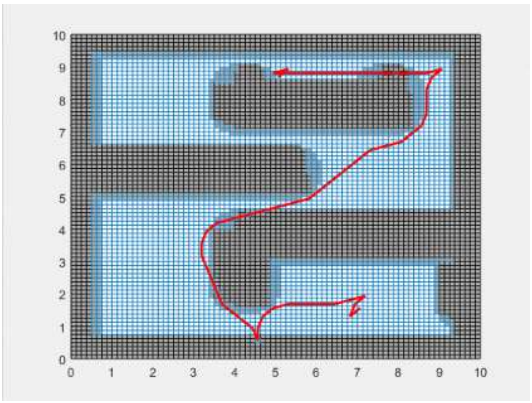
(a) Jednostavna prepreka



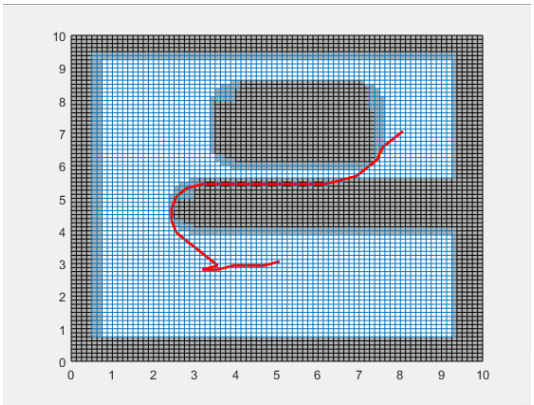
(b) Prepreka oblika slova 'U'



(c) Složene prepreke 1

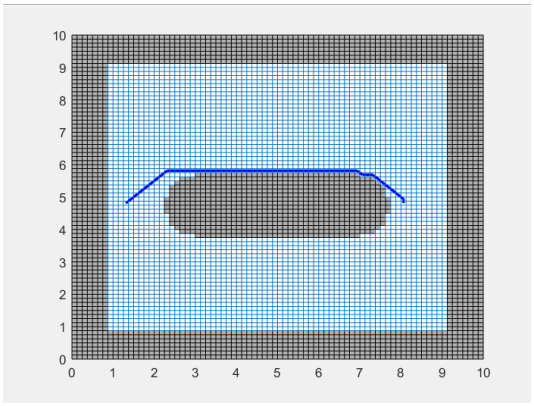


(d) Složene prepreke 2

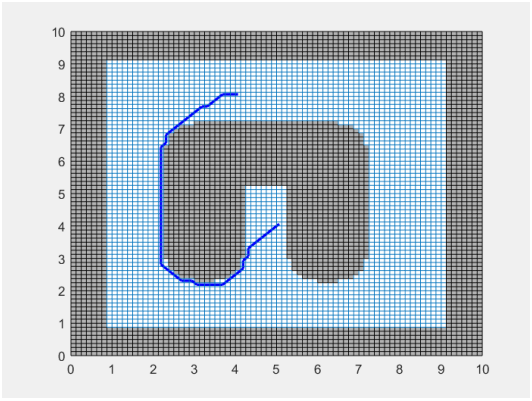


(e) Prepreke s uskim prolazom

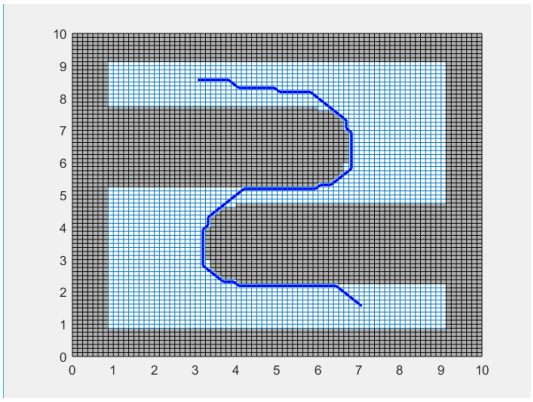
Slika 6.3. Prikaz putanje pronađenih prilagođenim A* algoritmom s različitim preprekama i teretom oblika slova 'T'



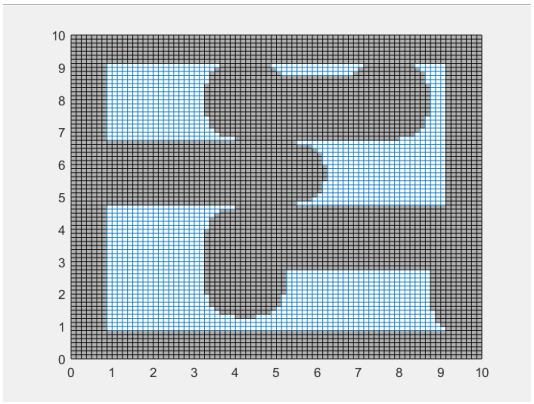
(a) Jednostavna prepreka



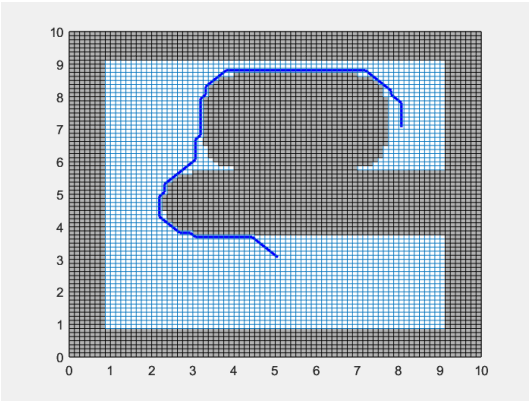
(b) Prepreka oblika slova 'U'



(c) Složene prepreke 1



(d) Složene prepreke 2

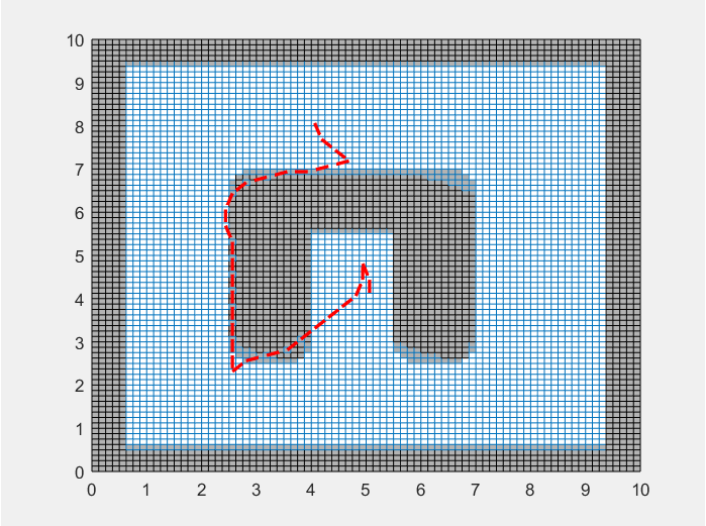


(e) Prepreke s uskim prolazom

Slika 6.4. Prikaz putanje pronađenih A* algoritmom s različitim preprekama

Kod prikaza putanja pronađenih korištenjem prilagođenog A* algoritma primjećuje se kako na pojedinim dijelovima putanja prividno prolazi kroz zauzeti prostor. Navedena pojava rezultat je prikaza svih 16 slojeva na jednom prikazu. Prostor koji je za jednu vrijednost orijentacije zauzet za različitu vrijednost orijentacije ne mora biti. Zbog navedenog putanja prividno prolazi kroz zauzeti prostor dok u stvarnosti to nije slučaj. Putanje dobivene prilagođenim A* algoritmom sadrže dijelove koji su šiljasti i koje kod klasičnog A* algoritma ne pronalazimo. Šiljasti dijelovi uglavnom se nalaze u dijelovima prostora u kojim se od robota očekuje da skreće ili da prilagodi orijentaciju tereta kako bi se postiglo efikasnije kretanje. U prilagođenom obliku A* algoritma pretražuju se samo dva susjedna orijentacijska sloja zbog čega robot mijenja smjer kretanja prema unaprijed ili unatrag kod oštrih skretanja. Kada robot ne bi imao mogućnost kretanja unatrag, zahtijevao bi puno veći prostor kako bi postigao očekivano skretanje.

Putanje prikazane na slikama 6.2. i 6.3. međusobno ne odstupaju značajno osim u slučaju kod složenog oblika prepreka verzije dva gdje algoritam za teret oblika slova 'T', zbog ograničenja konfiguracijskog prostora, pronalazi drugačiju putanju. Najveća odstupanja vidljiva su na području prostora gdje robot skreće zbog različitih oblika tereta,a time i konfiguracijskog prostora. Teret oblika slova 'T' kod kretanja blizu prepreka zahtijeva promjene orijentacije kako bi se što bolje prilagodio svojim nepravilnim oblikom, a pritom ostvario putanju s najmanjom cijenom. S obzirom na putanje klasičnog A* algoritma prikazane na slici 6.4., putanje dobivene pomoću prilagođenog A* algoritma mnogo su nepravilnije. Navedena pojava je očekivana i posljedica je ,osim već navedene prilagodbe orijentacije pri skretanju, zadane vrijednosti početne ili ciljne orijentacije. Pri definiranju početne orijentacije koja nije usmjerena prema ciljnoj poziciji robotu treba nekoliko iteracija kako bi postigao smjer kretanja prema željenom cilju. U radu je u svim slučajevima korištena jednaka početna orijentacija vrijednosti 11, odnosno 247.5 °. Na slici 6.5. prikazana je putanja za robot s pravokutnim teretom, ali različitom početnom orijentacijom od primjera na slici 6.2.b Putanja s novom orijentacijom vidno je različitog oblika, zbog čega se i cijene putanja razlikuju u takvom slučaju. Zbog navedenog, bitno je voditi računa pri zadavanju početne orijentacije.



Slika 6.5. Promijenjena početna orijentacija

Usprkos boljim rezultatima, kada se gleda vrijeme potrebno za pronalazak putanje od početne do ciljne pozicije korištenjem klasičnog A* algoritma, iznos cijene je za prilagođeni A* algoritam kod tereta pravokutnog oblika bolji u svim slučajevima. Kod tereta oblika slova 'T' prilagođeni A* algoritam u slučajevima gdje prepreke imaju uske prolaze daje bolje rezultate od klasičnog A* algoritma. Takav je rezultat očekivan i vidljiv samom usporedbom izgleda putanja na slikama 6.2., 6.3. i 6.4. Usporedba cijena za pojedine oblike prepreka i tereta za prilagođeni A* algoritam je prikazana u tablici 6.4., a za klasični A* algoritam u tablici 6.5. Kod prilagođenog A* algoritma iznos cijene je manji za oblik tereta pravokutnika za sve oblike prepreka. Takav rezultat je posljedica složenog konfiguracijskog prostora za teret oblika slova 'T' zbog čega robot češće mora prilagođavati orijentaciju kretanja ako se kreće uz prepreku. Također kod prilagodbe orijentacije pri skretanju veći broj iteracija mu je potreban kako bi prilagodio svoj kompleksni oblik. Pravokutni teret proširuje prepreke stvarajući oblike koji ne odstupaju previše od početne prepreke. To nije slučaj za teret oblika slova 'T' gdje proširene prepreke poprimaju izrazito nepravilne oblike s mnogo ispupčenja. Navedena pojava je vidljiva na slikama 5.3. i 5.4.

Oblik prepreka / Oblik tereta	Jednostavna	'U'	Složene 1	Složene 2	Uski prolaz
Pravokutnik	7.14	10.01	15.61	16.59	10.92
'T'	7.84	10.75	16.33	20.49	11.23

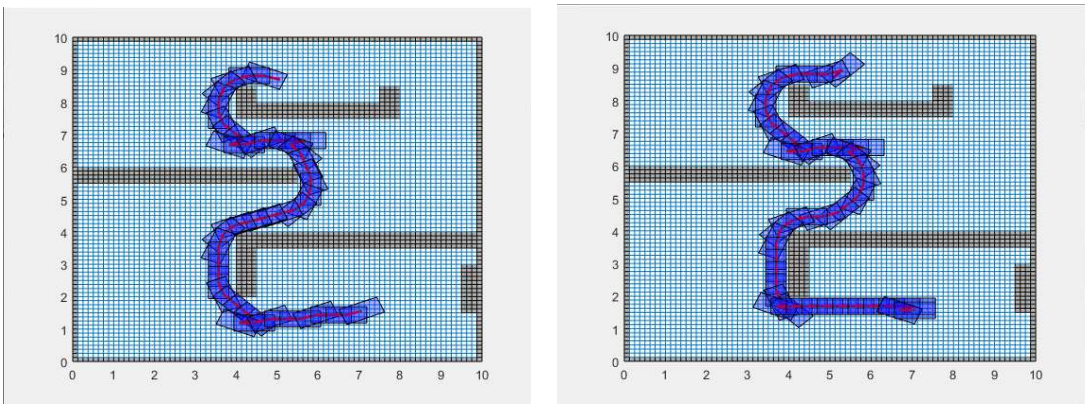
Tablica 6.4. Cijena pronalaska putanje ovisno o obliku prepreka i tereta za prilagođeni A* algoritam

Oblik prepreka	Jednostavna	'U'	Složene 1	Složene 2	Uski prolaz
	7.65	10.52	15.76	/	14.05

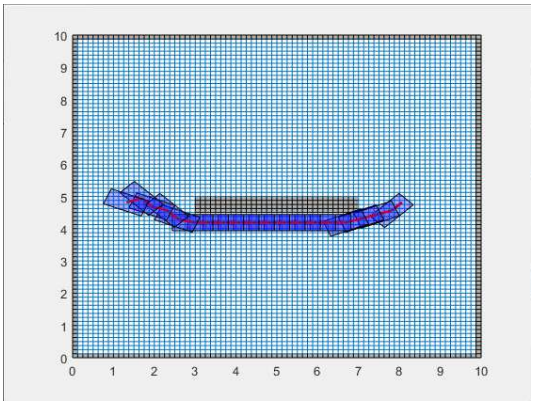
Tablica 6.5. Cijena pronalaska putanje ovisno o obliku prepreka za A* algoritam

Kako bi se bolje vizualizirala prilagodba putanje robota s određenim teretom u definiranom prostoru i promjene pri pojedinim orijentacijama na slikama 6.7. i 6.8. je prikazan otisak robota u svakoj pojedinoj točci pronađene putanje. Na slici 6.7. je prikazan otisak pravokutnog tereta koji je zarotiran ovisno o orijentaciji koju postiže u određenom čvoru, a na slici 6.8. je za jednake slučajeve prikazan teret oblika slova 'T'. Prikazane putanje i otisci robota dokazuju uspješnost prilagođenog A* algoritma da se s teretom nepravilnog oblika prilagođava preprekama na koje nailazi pri kretanju od zadane početne do ciljne pozicije.

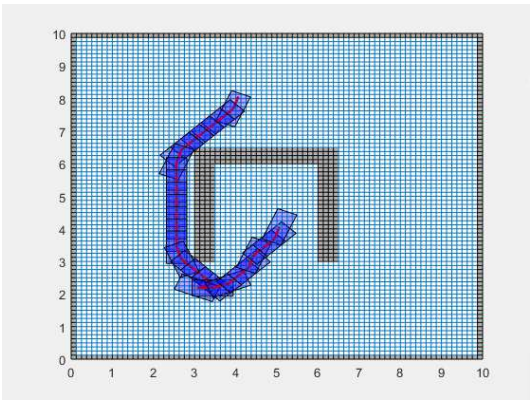
Vizualizacija pomoću otiska robota omogućuje i lakšu reprezentaciju promjene putanje i orijentacije za različite vrijednosti početne i ciljne orijentacije. Navedena pojava prikazana je na slici 6.6.



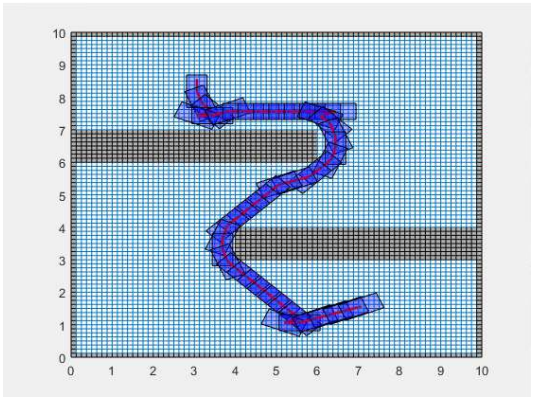
Slika 6.6. Prikaz putanje i otiska u prostoru s različitim početnim i ciljnim orijentacijama za pravokutni teret



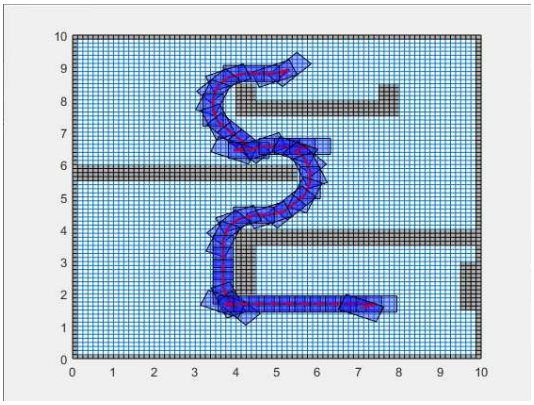
(a) Jednostavna prepreka



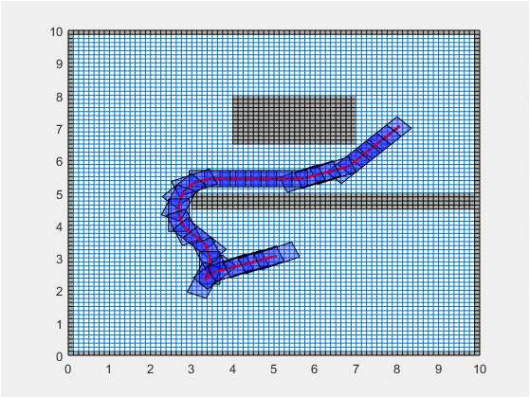
(b) Prepreka oblika slova 'U'



(c) Složene prepreke 1

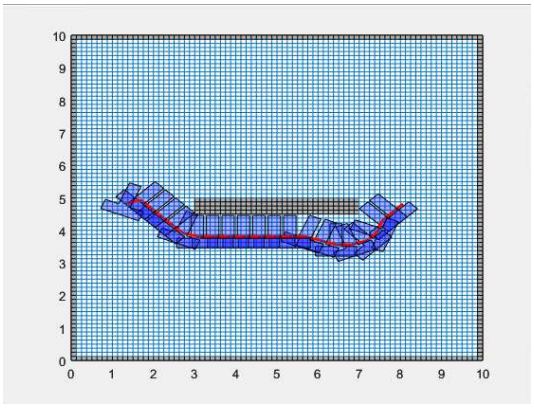


(d) Složene prepreke 2

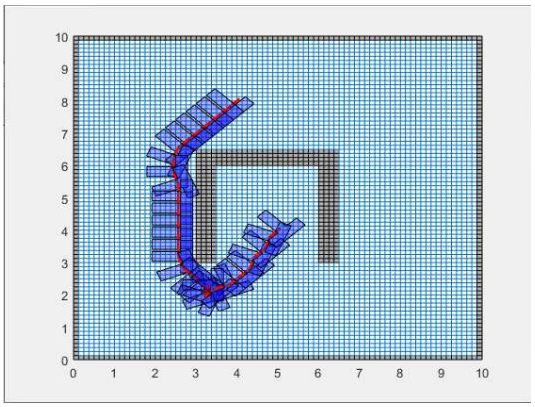


(e) Prepreke s uskim prolazom

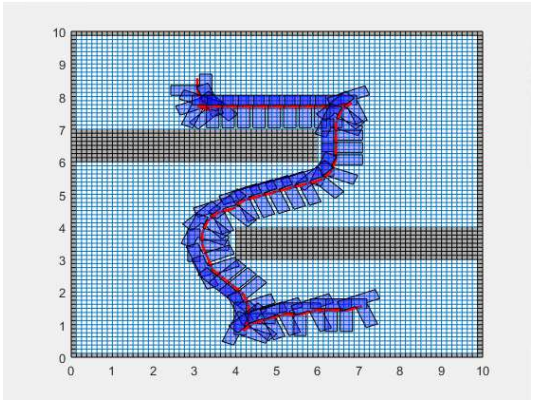
Slika 6.7. Prikaz putanje i otiska u prostoru s različitim preprekama i teretom oblika pravokutnika



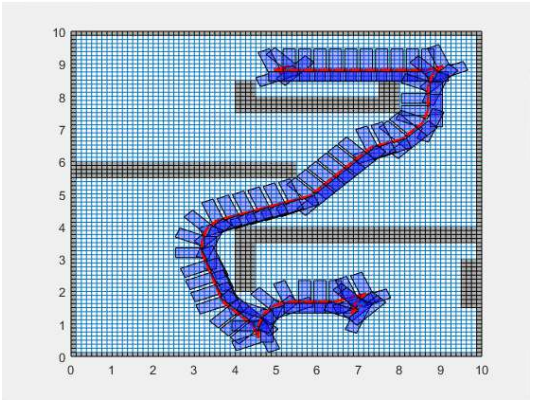
(a) Jednostavna prepreka



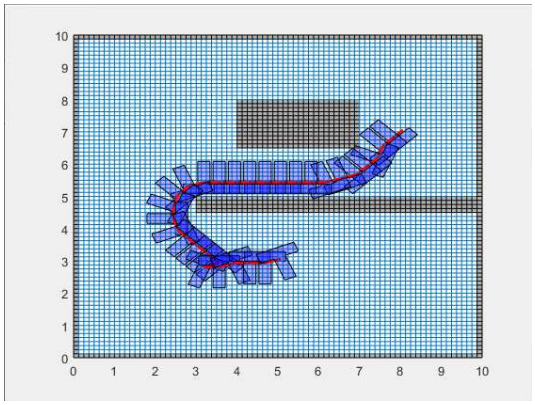
(b) Prepreka oblika slova ’U’



(c) Složene prepreke 1



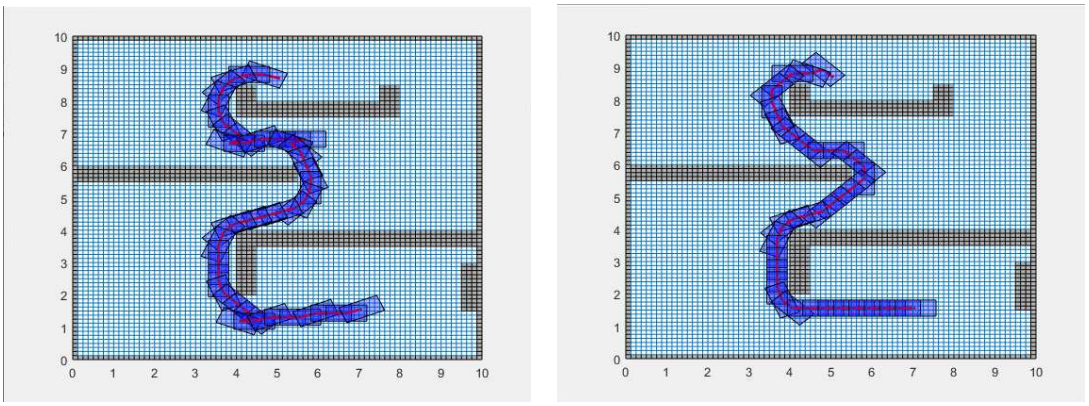
(d) Složene prepreke 2



(e) Prepreke s uskim prolazom

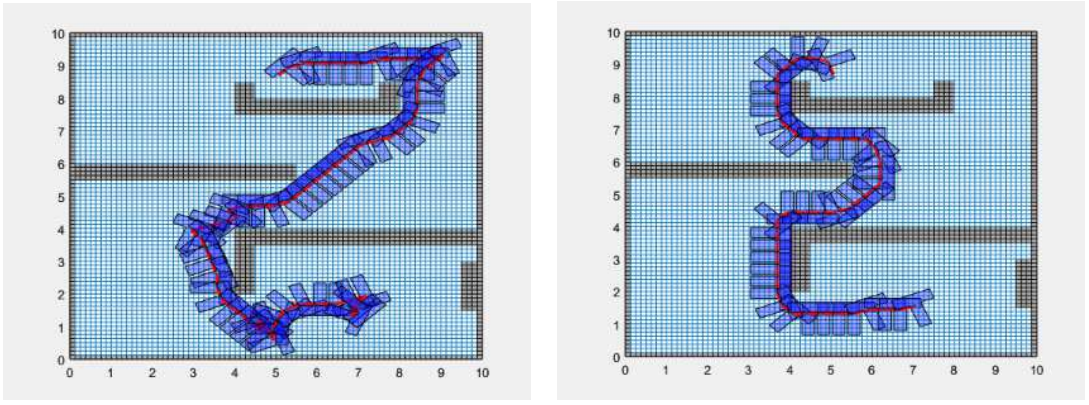
Slika 6.8. Prikaz putanje i otiska u prostoru s različitim preprekama i teretom oblika slova ’T’

U ovom radu pretražuju se samo dva susjedna sloja kako bi se uračunala kinematička ograničenja robota, ali testirani su i slučajevi kada se pretražuju četiri susjedna sloja. U slučaju kada se pretražuju četiri susjedna sloja robot pri kretanju od jednog čvora da sljedećeg može promijeniti orijentaciju za makinalno 45°. Time se robotu omogućuju oštrija skretanja kada je to potrebno što možemo vidjeti na slikama 6.9. i 6.10. Na slikama 6.9.a i 6.10.a prikazane su putanje kada se pretražuju dva susjedna sloja, a na slikama 6.9.b i 6.10.b kada se pretražuju četiri susjedna sloja. Vrijednost cijene za pravokutni oblik tereta kada se pretražuju samo dva susjedna sloja iznosi 16.71 dok kod pretraživanja četiri susjedna sloja iznosi 14.09. Za teret oblika slova 'T' vrijednosti cijene za dva susjedna sloja iznosi 21.08, a za četiri 16.07. Pretraživanje četiri susjedna orijentacijska sloja omogućava pronalazak kraćih putanja zbog toga što se željena promjena smjera može ostvariti u manjem broju iteracija algoritma i na manjoj površini prostora nego što je to slučaj kod pretraživanja dva susjedna sloja.



(a) Dva susjedna sloja **(b)** Četiri susjedna sloja

Slika 6.9. Prikaz putanje i otiska u prostoru za pravokutni teret



(a) Dva susjedna sloja **(b)** Četiri susjedna sloja

Slika 6.10. Prikaz putanje i otiska u prostoru za teret oblika slova 'T'

7. Zaključak

Mobilni roboti, kao autonomni sustavi, sve više postaju dio svakodnevice. Prisutni su u raznim djelatnostima gdje je prilagodljivost različitim uvjetima okoline bitna. Glavni izazov u postizanju autonomnosti i prilagodljivosti u okolini predstavlja planiranje putanje. Klasične metode planiranja putanje, iako učinkovite, često ne uzimaju u obzir posljedice koje različite orijentacije robota imaju na kretanje i stabilnost. Ovaj diplomski rad predstavio je prilagođeni A* algoritam dizajniran s ciljem rješavanja specifičnih izazova s kojima se suočavaju mobilni roboti koji nose nepravilan teret kod kretanju u okolini. Uključivanjem orijentacije u proces planiranja putanje, prilagođeni A* algoritam nudi rješenje koje mobilnim robotima omogućuje navigaciju u složenim okruženjima s većom učinkovitošću i sigurnošću. Prikazivanjem i analizom dobivenih rezultata je potvrđeno da oblik tereta značajno utječe na performanse mobilnih robota u pogledu planiranja putanje. Modificirani pristup poboljšava sposobnost robota da uzme u obzir svoja fizička ograničenja tijekom navigacije. To dovodi do optimiziranih putanja koje poboljšavaju upravljanje i učinkovitost kod robota s nepravilnim teretom. Rezultati pokazuju da roboti koji nose teret pravilnog oblika imaju bolje rezultate u pogledu vremenske učinkovitosti. Njihova predvidljiva raspodjela opterećenja i minimalan učinak na orijentaciju omogućuju glatku i bržu navigaciju u složenim okruženjima. Roboti s nepravilnim teretom daju lošije rezultate, ali u složenijim okruženjima bolje od klasičnog A* algoritma. Time se potvrđuje postignuta učinkovitost kod planiranja putanje za robote s nepravilnim teretom. Kada se uzme u obzir ukupna cijena puta, klasični A* algoritam i dalje se pokazuje kao optimalan zato što generira kraće putanje, ali takav je rezultat očekivan zbog razlike u konfiguracijskom prostoru. Prilagođeni A* algoritam, koji integrira orijentaciju u proces planiranja, nudi poboljšane performanse za robote koji nose nepravilne terete poboljšavajući učinkovitost i upravljanje. Ovi rezultati naglašavaju važnost odabira odgovarajućeg algoritma na temelju specifičnih zahtjeva zadatka i uvjeta okoline.

Literatura

[1] F. Staals, “Minimum enclosing circle”, https://www.mathworks.com/matlabcentral/fileexchange/13389-minimum-enclosing-circle, 2025., retrieved February 13, 2025.

[2] IEEE Standards Association, “Ieee standard for robot map data representation for navigation”, *1873-2015 IEEE Standard for Robot Map Data Representation for Navigation*, str. 1–54, 2015. https://doi.org/10.1109/IEEESTD.2015.7300355

[3] S. Yu, C. Fu, A. K. Gostar, i M. Hu, “A review on map-merging methods for typical map types in multiple-ground-robot slam solutions”, *Sensors*, sv. 20, br. 23, 2020. https://doi.org/10.3390/s20236988

[4] J. Ginés, F. Martín, V. Matellán, F. J. Lera, i J. Balsa, “3d mapping for a reliable long-term navigation”, u *ROBOT 2017: Third Iberian Robotics Conference*, A. Ollero, A. Sanfeliu, L. Montano, N. Lau, i C. Cardeira, Ur. Cham: Springer International Publishing, 2018., str. 283–294.

[5] H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki, i S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, 2005.

[6] E. Fernandes, P. Costa, J. Lima, i G. Veiga, “Towards an orientation enhanced astar algorithm for robotic navigation”, u *2015 IEEE International Conference on Industrial Technology (ICIT)*, 2015., str. 3320–3325. https://doi.org/10.1109/ICIT.2015.7125590

[7] Y. Shi, S. Huang, i M. Li, “An improved global and local fusion path-planning algorithm for mobile robots”, *Sensors*, sv. 24, br. 24, 2024. https:

//doi.org/10.3390/s24247950

[8] I. Petrović i I. Marković, “Planiranje putanje gibanja mobilnog robota”. [Mrežno]. Adresa: https://www.fer.unizg.hr/_download/repository/mr-p08.pdf

[9] C. Zhi, “Research on path planning of mobile robot based on a* algorithm”, *International Journal of Engineering Research and*, sv. V8, 11 2019. https://doi.org/10.17577/IJERTV8IS110186

[10] M. DJakulovic (Seder), “Planiranje gibanja autonomnih mobilnih robota u dinamickim i nepoznatim unutarnjim prostorima”, doktorska disertacija, Fakultet elektrotehnike i računarstva, 2010.

[11] I. Petrović i I. Marković, “Graph search and sampling-based path planning”. [Mrežno]. Adresa: https://www.fer.unizg.hr/_download/repository/amr-graph-sampling.pdf

[12] F. Duchoň, A. Babinec, M. Kajan, P. Beňo, M. Florek, T. Fico, i L. Jurišica, “Path planning with modified a star algorithm for a mobile robot”, *Procedia Engineering*, sv. 96, str. 59–69, 2014., modelling of Mechanical and Mechatronic Systems. https://doi.org/https://doi.org/10.1016/j.proeng.2014.12.098

[13] B. Lau, C. Sprunk, i W. Burgard, “Efficient grid-based spatial representations for robot navigation in dynamic environments”, *Robotics and Autonomous Systems*, sv. 61, br. 10, str. 1116–1130, 2013., selected Papers from the 5th European Conference on Mobile Robots (ECMR 2011). https://doi.org/https://doi.org/10.1016/j.robot.2012.08.010

Sažetak

Planiranje putanje mobilnog robota s teretom nepravilnog oblika

Ana Rusković

Mobilni roboti postali su neizostavan dio svakodnevice sa zadaćom autonomne navigacije u složenim okruženjima. Nepravilan oblik tereta kojeg roboti nose može značajno utjecati na njihovo kretanje i planiranje putanje. Tradicionalni algoritmi, poput klasičnog A* algoritma, usmjereni su na minimiziranje cijene puta u pogledu udaljenosti, ali ne uzimaju u obzir orijentaciju robota. Ovaj diplomski rad predlaže prilagodbu A* algoritma uzimajući u obzir orijentaciju robota prilikom planiranja putanje, optimizirajući vrijeme i cijenu navigacije. Provedene su simulacije kako bi se procijenila izvedba predloženog algoritma, a rezultati su pokazali da su roboti s pravilnim oblicima opterećenja imali najbolje rezultate u pogledu vremenske učinkovitosti. Međutim, prilagođeni A* algoritam dao je minimalne troškove putanje kod tereta s nepravilnim oblikom u složenim okruženjima. Ovaj rad prikazuje važnost kompromisa između različitih strategija planiranja putanje i pruža uvid u optimizaciju navigacije robota u različitim uvjetima.

Ključne riječi: mobilni robot; konfiguracijski prostor; planiranje putanje; otisak; prepreka; A* algoritam; orijentacija; cijena puta; čvorovi

Abstract

Path planning of mobile robots carrying any-shape weights

Ana Rusković

Mobile robots have become an indispensable part of of everyday life with the task of autonomous navigation in complex environments. The irregular shape of the load carried by the robots can significantly affect their movement and path planning. Traditional algorithms, such as the classic A* algorithm, focus on minimizing the path cost in terms of distance, but do not take into account the orientation of the robot. This thesis proposes an adaptation of the A* algorithm to take into account the orientation of the robot when planning the path, optimizing the time and cost of navigation. Simulations were conducted to evaluate the performance of the proposed algorithm, and the results showed that robots with regular load shapes had the best results in terms of time efficiency. However, the adapted A* algorithm gave the minimum path cost for irregularly shaped loads in complex environments. This paper shows the importance of trade-offs between different path planning strategies and provides insight into optimizing robot navigation under different conditions.

Keywords: mobile robot; configuration space; path planning; footprint; obstacle; A* algorithm; orientation; path cost; nodes

Privitak A: Funkcija za izračun opisane kružnice i njezinih parametara [1]

```
function [Xcenter, Ycenter, R] = SmallestEnclosingCircle(X, Y, Xouter, Youter)
% [Xcenter, Ycenter, R] = SmallestEnclosingCircle(X, Y)
% Purpose:
% Calculate the Smallest Enclosing Circle of a set of points
% Input:
% X-array with X coordinates of the points, size: 1 x nr_points
% Y-array with Y coordinates of the points, size: 1 x nr_points
% Input used for recursive use only:
% Xouter-array with 1, 2 or 3 X coordinates of outermost points, size: 1 x nr_outer_points
% Youter-array with 1, 2 or 3 Y coordinates of outermost points, size: 1 x nr_outer_points
% Output:
% Xcenter-X coordinate of smallest enclosing circle
% Ycenter-Y coordinate of smallest enclosing circle
% R-radius of smallest enclosing circle
% History:
% 14-Dec-2006 creation by FSta, based on an example by Yazan Ahed (yash78@gmail.com),
%who based his code on a Java applet by Shripad Thite (http://heyoka.cs.uiuc.edu/~thite/mincircle/)
%Initialize Xouter, Youter, nr_outer_points
if nargin < 4
    Xmin = min(X);
    Ymin = min(Y);
    size_max = max(max(X) - Xmin, max(Y) - Ymin);
    Xouter = NaN; %init
    Youter = NaN; %init
    nr_outer_points = 0;
else
    nr_outer_points = size(Xouter, 2);
end; %if nargin < 4
switch nr_outer_points
    case 0
        Xcenter = (max(X) + Xmin) / 2;
        Ycenter = (max(X) + Ymin) / 2;
        R = 0;
    case 1
        Xcenter = Xouter(1);
```

```
        Ycenter = Youter(1);
        R = 0;
    case 2
        Xcenter = (Xouter(1) + Xouter(2)) / 2;
        Ycenter = (Youter(1) + Youter(2)) / 2;
        R = sqrt((Xouter(1) - Xcenter)^2 + (Youter(1) - Ycenter)^2);
    case 3
        Xcenter = (Xouter(3)^2 * (Youter(1) - Youter(2)) + (Xouter(1)^2 + (Youter(1) - Youter(2)) *
             (Youter(1) - Youter(3))) * (Youter(2) - Youter(3)) + Xouter(2)^2 * (-Youter(1) +
             Youter(3))) / (2 * (Xouter(3) * (Youter(1) - Youter(2)) + Xouter(1) * (Youter(2) -
             Youter(3)) + Xouter(2) * (-Youter(1) + Youter(3))));
        Ycenter = (Youter(2) + Youter(3)) / 2 - (Xouter(3) - Xouter(2)) / (Youter(3) - Youter(2)) *
             (Xcenter - (Xouter(2) + Xouter(3)) / 2);
        R = sqrt((Xouter(1) - Xcenter)^2 + (Youter(1) - Ycenter)^2);
        return;
end; %switch nr_outer_points
for i = 1:size(X, 2)
    if (X(i) - Xcenter)^2 + (Y(i) - Ycenter)^2 > R^2
            if isempty(intersect([Xouter' Youter'], [X(i)' Y(i)'], 'rows'))
                Xouter(nr_outer_points + 1) = X(i);
                Youter(nr_outer_points + 1) = Y(i);
                [Xcenter, Ycenter, R] = SmallestEnclosingCircle(X(1:i), Y(1:i), Xouter, Youter);
            end; %if isempty(intersect([Xouter Youter], [X(i) Y(i)], 'rows'))
    end; %if (X(i) - Xcenter)^2 + (Y(i) - Ycenter)^2 > R^2
end; %for i
return;
```