**RĪGAS TEHNISKĀ UNIVERSITĀTE**

Datorzinātnes un informācijas tehnoloģijas fakultāte

Lietišķo datorsistēmu institūts

**Domeniks Delvers**

bakalaura akadēmisko, pirmā līmeņa studiju programmas

students, stud. apl. nr. 201RDB298

**PĀRKLĀJUMA CEĻA PLĀNOŠANAS ALGORITMU SALĪDZINĀJUMS**

**PUTEKĻSŪCĒJU ROBOTIEM**

**BAKALAURA DARBS**

Zinātniskais vadītājs Doktors, Profesors

Agris Ņikitenko

RĪGA 2022

**RĪGAS TEHNISKĀ UNIVERSITĀTE**

**DATORZINĀTNES UN INFORMĀCIJAS TEHNOLOĢIJAS FAKULTĀTE**

<institūta nosaukums>

<struktūrvienības nosaukums>

**<Noslēguma darba no saraksta: bakalaura darbs, maģistra darbs, diplomprojekts, kvalifikācijas darbs> izpildes lapa**

Noslēguma darba autors:

students(-e) <vārds, uzvārds>\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(paraksts, datums)

Noslēguma darbs ieteikts aizstāvēšanai:

Zinātniskais vadītājs:

<zinātniskais grāds, amats, vārds, uzvārds> \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(paraksts, datums)

Anotācija

Putekļsūcēj-robots, simulācija, ceļa plānošanas algoritms

Mūsdienās mājas tīrīšanas roboti paliek arvien izplatītāki, un īpaši – putekļsūcēj-roboti. Viena no īpašībām, kas raksturo robotu ir tā tīrīšanas ātrums, tāpēc ražotāji mēģina izstrādāt pēc iespējas efektīvākus algoritmus, lai robots spētu iztīrīt visu dzīvokļa platumu pēc iespējas mazākā laikā. Šajā darbā tiek apskatīti vairāki tādi algoritmi, kā arī simulēti virtuālajā vidē un salīdzināti dažādos apstākļos.

Darbā ir X lappušu, Y attēlu, Z tabulu, W pielikumu un Q izmantoto avotu skaits.

Abstract

Nowadays, home cleaning robots are becoming more and more popular, particularly vacuum-cleaning robots. One of the characteristics that defines a robot is its cleaning speed, so manufacturers are trying to develop as efficient algorithms as possible so that the robot can clean the entire width of the apartment in the shortest possible time. In this work, several such algorithms are examined and simulated in a virtual environment and compared under different conditions.

Satura rādītājs

[Anotācija 2](#_Toc128468845)

[Abstract 3](#_Toc128468846)

[Satura rādītājs 4](#_Toc128468847)

[Ievads 5](#_Toc128468848)

[1. Iekštelpas vides procedurāla ģenerēšana 7](#_Toc128468849)

[1.1. Vides skeleta veidošana 7](#_Toc128468850)

[1.1. Durvju izvietojums 8](#_Toc128468851)

[1.2. Mēbeļu un šķēršļu ģenerēšanā 9](#_Toc128468852)

[2. Testu veicošais aģents 11](#_Toc128468853)

[2.1. Aģenta struktūra 11](#_Toc128468854)

[2.2. Aģenta vadība 12](#_Toc128468855)

[3. Pilna pārklājuma ceļa plānošana 13](#_Toc128468856)

[3.1. Pilna pārklājuma ceļa plānošanas algoritmu veidi 13](#_Toc128468857)

[3.2. Vides dekompozīcija 14](#_Toc128468858)

Ievads

Pēdējos gados ir palielinājusies interese pret pilna pārklājuma ceļa meklēšanas algoritmu attīstīšanu robotizēto sistēmu un mākslīgā intelekta nozarēs. Iespēja efektīvi izbraukt un pārklāt dažādas kompleksas, dinamiskas vides ir ļoti svarīga vairākiem uzdevumiem: glābšanas operācijām, drošības uzraudzībai, kartēšanai, vides monitoringam un īpaši tīrīšanai. Mājas tīrīšanas robotu tirgū jau no 2014 gada ir novērojama stabila izaugsme. Palielinoties interesei pēc gudrās mājas tehnoloģijas un automatizācijas, pieprasījums pēc mājas tīrīšanas robotiem arvien aug. Īpaša tirgus izaugsme tika novērota COVID – 19 pandēmijas laikā. Sakarā ar pandēmiju daudzi cilvēki bija spiesti pavadīt vairāk laika mājās, tāpēc palielinājās interese pēc produktiem, kas ļauj saturēt dzīvokļus tīrākus. Saskaņā ar “*Research and Markets*” pētījumu, starp 2021. gadu un 2026 gadu, var sagaidīt 20% salikto gada pieauguma tempu (*Compound Annual Growth Rate*). Zinātniskais raksts pamato tik strauju izaugsmi ar palielinātu pieprasījumu pēc inovatīviem mājas tīrīšanas veidiem, kā arī gudrās mājas tehnoloģijas izplatību. Mājas tīrīšanas robotu tirgū ir daudz konkurējošo ražotāju, piemēram: “*iRobot*”, “*Dyson*”, “*LG Electronics*”, “*Ecovacs* *Robotics*” un “*Samsung* *Electronics*”. Augot pieprasījumam pēc mājas tīrīšanas robotiem, ražotāji cenšas uzlabot savu robotu efektivitāti, lai piesaistītu vairāk klientu.

Īpaši būtiski robotu efektivitāti ietekmē tā programmatūra: navigācijas algoritms, kartēšanas algoritms un ceļa plānošanas algoritms. Lielākā daļā tieši no ceļa plānošanas algoritma ir atkarīgs, cik ātri un kvalitatīvi robots paveiks savu uzdevumu. Algoritmam ir jānodrošina pilns telpas pārklājums, šķēršļu apbraukšana, kā arī jāatrod ceļš ar minimālo garumu un mazāku pagriezienu skaitu. Lai veicinātu pilnu pārklājumu ražotāji izmanto dažādas metodes, kuras savukārt ir atkarīgas no vienlaicīgas lokalizācijas un kartēšanas (*SLAM*) sistēmas precizitātes. Mūsdienīgie tīrīšanas roboti izmanto heiristiskas metodes pārklājuma ceļa meklēšanai vairāku iemeslu dēļ. Pirmkārt *SLAM* sistēmu neprecizitātes dēļ ir grūti precīzi izbraukt iepriekš ieplānoto trajektoriju, kā arī nav iespējams pietiekami precīzi nokartēt vidi. Otrkārt visi mūsdienās pastāvošie pilna pārklājuma algoritmi ir nepiemēroti mājas apstākļiem.

Šodien neviens ražotājs nepiedāvā efektīvu risinājumu klientiem, kuri vēlās izmantot savās mājās vairākus tīrīšanas robotus vienlaicīgi. Mūsdienas tīrīšanas robotiem nav sistēmu, kas nodrošinātu efektīvu vairāku aģentu sadarbību kopīgas problēmas risināšanai.

Ceļu plānošanas uzdevums dzīvokļos un mājās ir izaicinošs vides nenoteiktības un šķēršļu klātbūtnes dēļ. Viens veids, kā efektīvi meklēt risinājumu šīm izaicinājumam, ir algoritma testēšanas laikā izmantot procedurāli ģenerētas vides, kurās vides plāns un šķēršļi tiek ģenerēti algoritmiski, nevis tiek iepriekš definēti. Tas ļauj izveidot ļoti dažādas un sarežģītas vides ar lielu variāciju. Šādās vidēs var testēt dažādu ceļu plānošanas algoritmu veiktspēju. Šajā darbā tiek salīdzināti dažādi pilna pārklājuma ceļa meklēšanas algoritmi procedurāli ģenerētās telpās, ar fokusu uz mājas tīrīšanas robotiem. Darbā tiek salīdzināti dažādi algoritmi pēc to ātruma, efektivitātes un izturības.

Darba mērķis ir simulēt vairāku aģentu pārklājuma problēmas risinošo algoritmu virtuālajā vidē, pie dažādiem apstākļiem un izpētīt piedāvātā risinājuma efektivitāti. Lai veiksmīgi sasniegtu šo mērķi tika izvirzīti sekojoši darba uzdevumi:

1. Sagatavot virtuālo vidi eksperimentu veikšanai. Šīm nolūkam tika izmantota “Unity 3D”, kur procedurāli tiek ģenerēti dažādi dzīvokli.
2. Īstenot istabas apiešanas algoritmu, ar kuru palīdzību var veiksmīgi apiet iepriekš ģenerētas vides.
3. Savākt datus un normalizēt tos, lai veiktu pilnvērtīgus secinājumus.

Darba pirmajā nodaļā ir aprakstīts iekštelpas vides procedurālas ģenerēšanā “Unity3D” vidē. Darba otrajā nodaļā tiek aprakstīti testētie pilnās apiešanas algoritmi, kā arī pamatota to izvēle. Trešajā nodaļā tiek aprakstīts testēšanas, datu savākšanas un secināšanas process.

1. Teorijas pārskats
   1. Tradicionālas pārklājuma meklēšanas tehnikas
      1. Šūnu dekompozīcija

Šūnu dekompozīcija ir tehnika, kas sadala vidi vairākos nepārklājošās šūnās, kuras nākotnē varētu apceļot aģents. Viens no agrākajiem darbiem šajā nozarē pieder *Moravec* un *Elfes* kungiem (1985), kas pielietoja okupācijas režģa pamatā esošo pieeju robotu kartēšanai un navigācijai (High resolution maps from wide angle sonar). Vēlāk, zinātnieki *Choset* un *Pignon* (1997) piedāvāja bustrofedona dekompozīcijas metodi, kas sadala vidi šūnās, kas robotiem ir viegli šķērsojamas (Coverage Path Planning: The Boustrophedon Cellular Decomposition).

* + 1. Bustrofedona dekompozīcija

Bustrofedona dekompozīcija, ko ieviesa *Choset* un *Pignon* (1997), ir populāra šūnu dekompozīcijas pieeja ceļu plānošanai, kas nodrošina pilnīgu pārklājumu (Coverage Path Planning: The Boustrophedon Cellular Decomposition). Šī tehnika sadala vidi šūnās, kuras pēc tam aģents šķērso veicot “S” veida kustības. Pētnieki ir turpinājuši attīstīt un uzlabot šo pieeju, piemēram *Acar* (2002) darbā, kur tie piedāvāja pilnīgu ceļu plānošanas algoritmu robotiem ar ierobežotām sensoru spējam (Sensor-based Coverage of Unknown Environments: Incremental Construction of Morse Decompositions).

* + 1. Režģa bāzētās metodes

Uz režģi bāzētās pieejas diskretizē vidi, lai iegūtu vides režģi un pēc tam izmanto grafu bāzētos algoritmus pārklājuma nodrošināšanai. Piemēram, *Zelinsky* (1993) ieviesa viļņa frontes bāzēto pārklājuma ceļa meklēšanas algoritmu, kas izmanto režģa rūtiņas, kā pamatelementus vides izmeklēšanai (Planning Paths of Complete Coverage of an Unstructured Environment by a Mobile Robot). Vēlāk *Huang* (2001) piedāvāja tā saucamo “optimālo nejaušo pastaigu” režģa bāzētu pārklājuma ceļa meklēšanas algoritmu (Optimal line-sweep-based decompositions for coverage algorithms).

* + 1. Aptveroša koka metodes

Tehnikas, kuras pamatojās uz aptveroša koka metodi, veido koku, kas aptver visu vidi, un kuram izseko aģents, lai paveiktu pilnu vides pārklāšanu. *Gabriely* un *Rimon* (2001) ieviesa aptveroša koka pārklājuma (STC) algoritmu, kas veido aptverošo koku par režģa pamatā esošo vidi un garantē pilnīgu pārklājumu (Spanning-tree based coverage of continuous areas by a mobile robot). Šo pieeju ir turpmāk izpētījusi un paplašinājuši citi pētnieki, piemēram *Hazon* un  *Kaminka* (2005) darbā, kurā tie piedāvāja tiešsaistes algoritmu STC metodei (Redundancy, Efficiency and Robustness in Multi-Robot Coverage).

1.2. Apmācība ar pastiprinājumu vienīga aģenta pārklājuma ceļa plānošanai

1.2.1. Q – apmācība

*Engel* (2005) demonstrēja Q – apmācības potenciālu vienīga aģenta pārklājuma ceļa plānošanas uzdevuma risināšanā (Reinforcement learning with Gaussian processes). Pētnieks piemēroja Q – apmācības metodi lauksaimnieciskas vides autonomām izsmidzināšanas nolūkam un pieradīja tehnoloģijas efektivitāti.

1.2.2. Dziļie Q – tīkli

Dziļie Q – tīkli (DQN) paplašina tradicionālo Q – mācīšanas metodi, izmantojot dziļos neironu tīklus funkciju aproksimācijai. *Mnih* (2015) veiksmīgi pielietoja DQN “Atari” konsoles spēlēm (Human-level control through deep reinforcement learning). Lai gan DQN nav plaši pielietoti pārklājuma ceļa plānošanai, to veiksmīgs pielietojums komplicētās vidēs pierāda potenciālu pielietojamību.

1.2.3. Aktiera – kritiķa metodes

Aktiera – kritiķa metodes ir piemērotas pārklājuma ceļa plānošanai, kā redzams *Konda* un *Tsitsiklis* (2002) darbā, kas prezentēja aktiera – kritiķa algoritmu mācīšanas ar pastiprinājumu metodei (On actor-critic algorithms). Viņi pierādīja, ka algoritms spej mācīties un pielāgoties dažādām vidēm un apstākļiem.

1.2.4. Monte Carlo koka pārmeklēšana

Monte Carlo koka pārmeklēšana “MCTS” ir ceļa plānošanas algoritms, kas apvieno Monte Carlo simulācijas ar koka pārmeklēšanu. MCTS tika veiksmīgi pielietots vienīga aģenta pārklājuma ceļa plānošanai, kā to pierādījis *Coulom* (2006), kurš izmantoja MCTS spēlei “Go” (Efficient selectivity and backup operators in Monte-Carlo tree search).

1.3. Apmācība ar pastiprinājumu vairāku aģentu pārklājuma ceļa meklēšanai

1.3.1. Neatkarīgi mācekļi

Neatkarīga Q – mācīšanas metode tika pielietota vairāku aģentu pārklājuma ceļa plānošanai *Matignon* (2007) darbā, kas ieviesa sadarbības Q – mācīšanas algoritmu vairāku aģentu pārklājuma ceļa meklēšanai (Independent reinforcement learners in cooperative Markov games: A survey regarding coordination problems). Darbs pierāda, ka pieeja var efektīvi koordinēt vairākus robotus, risinot nenoteiktības dinamiskās vidēs.

1.3.2. Kopīgu darbību mācekļi

Kopīgu darbību mācības metode ņem vērā visu aģentu kopējās darbības mācību procesā. *Oliehoek* (2006) pielietoja kopīgu darbību mācības metodi vairāku aģentu pārklājuma ceļa meklēšanai, prezentējot decentralizētu algoritmu, kas ir balstīts uz vairāku aģentu Markova lēmumu procesiem (MMDP) (Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems). Pētnieks demonstrēja algoritma mērogojamību un izturību dažādos testēšanas scenārijos.

1.3.3. Centralizēta apmācība ar decentralizētu izpildi

1.3.4. Komunikācijas pamatotas pieejas

1. Iekštelpas vides procedurāla ģenerēšana

Procedurālā iekštelpu vides ģenerēšana ir tehnika, kas ļauj virtuāli veidot telpas un vides. Šī tehnika izmanto algoritmus un noteikumus, lai automātiski ģenerētu nejaušo telpas plānojumu, izmantojot programmatūru. Rezultējošās telpas var tikt izmantotas video spēlēs, simulācijās un citās interaktīvās lietojumprogrammās. Procedurāli ģenerēto vižu labums ceļa meklēšanas algoritmu testēšanai ir tas, ka procedurāli ģenerētas vides piedāvā lielāku variāciju telpu savstarpējam novietojumam, un šķēršļu novietojumam. Algoritmu testēšana telpās, kas pastāv reālajā dzīvē, ierobežo apstākļu variāciju, jo lielākoties dzīvokļu plānojumi balstās uz vienādiem arhitektūras un dizaina principiem, secīgi – tiem bieži ir līdzīgas īpašības. Savukārt robota efektivitāti neietekmē cik skaisti, vai ērti ir izvietotas mēbeles istabā, to ietekmē tikai iespēja precīzi un kvalitatīvi plānot savu ceļu jebkurā vidē. Šajā darbā, procedurālo vides ģenerēšanu var sadalīt trijos būtiskajos posmos: vides skeleta veidošana, durvju izvietošana un mēbeļu izvietošana.

* 1. Vides skeleta veidošana

Vides skelets, šī darba kontekstā ir primitīva vide, kas sastāv no vairākām ortogonālām istabām, kurām ir noteikti izmēri, izvietojums telpā un tips. Vides skeleta būvēšana ir būtiskākais solis procedurālā ģenerēšanā, jo tas nosaka kopējo vides formu un vides kopējo platību. Šajā algoritmā katra istaba ir istabas (*Room*) klases objekts, kas glabā informāciju par istabas izmēriem, istabas centra koordinātēm, istabas stūru koordinātēm, istabas tipu un istabas identifikācijas numuru. Jaunas istabas būvēšana sākās ar nejaušo istabas tipa izvēli, piemēram “virtuve” (*kitchen*). Pēc tipa izvēlēšanas, algoritms nejauši izvēlās jaunas istabas izmērus, ar ierobežojumiem, kuri ir atkarīgi no istabas tipa. Zinot istabas izmērus, algoritms izmeklē visas potenciālas izvietošanas iespējas. Izvietošanai ir daži ierobežojumi: istaba nedrīkst pārklāt vai krustot citu istabu, neviens no istabas punktiem nedrīkst atrasties zem nulles koordinātes x vai z asī. Pēc visu potenciālu izvietošanas punktu atrašanas, uz nejauši izvēlēta punkta tiek būvēta jauna istaba. Process atkārtojās tikmēr nebūs izveidots nejauši izvēlētais istabu skaits.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  |  | | --- | --- | |  |  | |  |  | |
| 1.att. Procedurāli ģenerēto vides skeletu piemēri |

* 1. Durvju izvietojums

Nākamais solis vides ģenerēšanā ir durvju ģenerēšana. Darba mērķim nav vajadzīgas fiziskas atveramas durvis, tāpēc to vietā ir noteikta platuma caurumi sienās, kas ļaus aģentam brīvi pārvietoties starp visām istabām. Lai sasniegtu visu istabu pieejamību, tiek veidots plānojumu reprezentējošs grafs, kur katrs grafa mezgls reprezentē vienu no istabām. Iegūtam grafam tiek atrasts mazākais pārklājošais koks (*minimum spanning tree*), lai noteiktu minimālu durvju skaitu un izvietojumu, kas nodrošinās iespēju aģentam iziet cauri visām ģenerētām istabām. Turpmāk ar nelielu varbūtību tiek ģenerētas vel dažas durvis, ar priekšroku lielākām istabām, lai nodrošinātu lielāku variāciju..

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| 2.att. durvju izvietošanas algoritma darbības princips pa soļiem | | | |

|  |  |
| --- | --- |
|  |  |
| 3.att. Procedurāli ģenerēto vižu piemēri ar durvīm | |

* 1. Mēbeļu un šķēršļu ģenerēšanā

Reālajā dzīvē dzīvokļos un mājās vienmēr ir mēbeles. Mēbeles kalpo dažādiem nolūkiem, bet tīrīšanas robotiem tie pirmkārt ir šķērslī, kas ir jāņem vērā plānojot apiešanas maršrutu. Lai pilnās apiešanas algoritmu testēšanas rezultāti būtu pilnvērtīgi, arī simulācijā ir jāiekļauj mēbeļu ģenerēšana. Lai izvietot mēbeles uzticami, ir jāņem vērā vairāki noteikumi. Pirmkārt, katrā istabā var atrasties tikai konkrēta tipa mēbelēs, piemēram guļamistabā obligāti jābūt gultai, bet virtuvē nevajadzētu būt tualetei. Istabas tips tiek noteikts vel pirmajā solī, skeleta ģenerēšanas laikā. “Sēklas”, kas tika nejauši ģenerētas ir objekti, kas satur informāciju arī par istabas tipu. Ģenerējot mēbeles algoritms pirmkārt atsaucās uz to informāciju, lai veidotu zināšanu bāzi: kādām mēbelēm obligāti jābūt istabā, kādas mēbeles var būt istabā un kādas mēbeles ģenerēt nedrīkst noteiktajā istabā.

Otrkārt, ir jāņem vērā arī savstarpējo novietojumu mēbelēm. Piemēram mēbeles nedrīkst stāvēt durvīm priekšā, jo tādā gadījuma aģents nevarēs izbraukt tām cauri. Turpmāk, mēbeles ir jāizvieto tā, lai nerādītu pārāk šauras vietas arī istabas iekšā, pretējā gadījumā testēšanas vidē būs nesasniedzami apgabali, un testa rezultātus nevarēs uzskatīt par pilnvērtīgiem. Turklāt jāņem vērā arī mēbeļu izvietošanas loģiskumu. Piemēram ja dzīvojamā istabā ir televizors, tad dīvānam jābūt orientētam pret televizoru.

1. Testu veicošais aģents

Aģents šajā simulācijā attiektos uz vakuuma tīrīšanas robotu, kurš tiek testēts. Aģents būtu atbildīgs par pārklājuma ceļu plānošanas algoritmu izpildi. Aģenta veiktspēja tiks novērtēta, pamatojoties uz to, cik efektīvi tas spēj pārklāt vidi un tīrīt to.

Eksistē dažādi faktori, kas var ietekmēt aģenta veiktspēju šajā simulācijā, piemēram, vides sarežģītība, pārklājuma ceļu plānošanas algoritmu efektivitāte un robotu spējas (pēdējais paliks nemainīgs visiem eksperimentiem). Simulācijas mērķis būtu noskaidrot visefektīvākos pārklājuma ceļu plānošanas algoritmus, lai uzlabotu tā veiktspēju un efektivitāti.

* 1. Aģenta struktūra

Aģentam jābūt spējīgam pārvietoties taisni un apgriezties ap savu asi, lai veiksmīgi sekot ieplānotam ceļam. Lai to paveiktu aģentam ir divi riteņi sānos, kas var neatkarīgi griezties vēlamā virzienā ar vēlamo ātrumu. Kontrolējot riteņu griešanas virzienu var veiksmīgi izpildīt visas nepieciešamās aģenta funkcijas. Griežot abus riteņus vienā virzienā aģents pārvietosies pa taisnā līnijā uz priekšu, jeb atpakaļgaitā. Griežot riteņus dažādos virzienos ar vienādu ātrumu, var paveikt tā rotāciju ap savu ass velējama virzienā. Aģentam nav nekādu sensoru, jeb atgriezeniskas saites, jo šajā darbā lokalizācijas un navigācijas problēmas nav apskatītas.

|  |
| --- |
|  |
| 4.att. Aģenta prototips ar četriem riteņiem |

* 1. Aģenta vadība

Aģenta vadība tiek nodrošinātā ar tā saucamo “*waypoint system*”, jeb ceļrāžu sistēmu. Idejiski robots nespēj pats izplānot ceļu, to dara atsevišķs algoritms, kas pārsūt robotam rindu ar komandām – kur ir jābrauc. Katra komanda no rindas ir punkts ar koordinātēm, un saņemot komandu robots tiecās uzbraukt virsū punktam vidē, kuram atbilst saņemtās koordinātes. Aģents paveic to divos soļos: pirmais solis ir apgriezties ap savu asi, kamēr viņš nebūs orientēts tieši pret intereses punktu; otrais solis ir braukt uz priekšu tikmēr aģenta koordinātes nesakritīs ar ceļrāža koordinātēm. Izpildot šos divus soļus, aģents var ķerties pie nākamās komandas izpildes.

Ceļrāžu rinda tiks ģenerēta pilnās apiešanas ceļa plānošanas algoritmu rezultātā. Ceļrāžu rindai jānodrošina, ka izbraucot pāri visiem ceļrāžiem, tiks sasniegta pilna vides pārklāšana.

1. Pilna pārklājuma ceļa plānošana

Pilna pārklājuma ceļa plānošana ir problēma, kurā meklē ceļu, kuru izbraucot, aģents pilnīgi pārklātu noteiktu reģionu vai vidi, neizlaižot nekādas daļas vai atstājot nekādas neapklātas zonas. Tas bieži ir svarīgs jautājums, piemēram, tīrīšanas, kartēšanas, pārbaudes un uzraudzības aplikācijās, kur svarīgi, lai robots pārklātu visu vides zonu rūpīgi un efektīvi.

Problēmas pilna pārklājuma ceļa plānošanai var risināt dažādos veidos, atkarībā no konkrētajām prasībām un ierobežojumiem. Neatkarīgi no izmantotās pieejas, pilna pārklājuma ceļa plānošanā bieži ir jāveic kompromisi starp pārklājuma efektivitāti, ceļa garumu un citiem faktoriem, piemēram, enerģijas patēriņu vai laika ierobežojumiem.

* 1. Pilna pārklājuma ceļa plānošanas algoritmu veidi

Mūsdienīgie mājas tīrīšanas roboti pašlaik neizmanto pilna pārklājuma ceļa meklēšanas algoritmus. Visi algoritmi, kas spēj ieplānot ceļu ar pilnu pārklājumu un nodrošina tieši īsāko iespējamo ceļu ir bāzēti uz vides režģi (*grid based*). Tā ir liela problēma mājas tīrīšanas robotu nozarē, jo atkarība no vides režģa neļauj lietot efektīvus algoritmus dzīvokļos. Tas ir saistīts ar to ka visos algoritmos kas ir bāzēti uz vides režģa, viena mezgla, jeb rūtiņas, izmērs ir definēts kā aģenta pārklājuma izmērs. Tas nozīmē ka mājas tīrīšanas robotam vienas rūtiņas platums būs vienāds ar robota platumu. Katra rūtiņa režģī var būt vai apmeklējama, vai šķērslis, kas būtiski nozīmē, ka neatkarīgi no šķēršļa izmēra, tā klātbūtne atzīmēs veselu rūtiņu kā neapmeklējamu, tāpēc pilnu pārklājumu nevarēs sasniegt. Arī otrādi, ja definēt režģa rūtiņas izmēru kā mazāka šķēršļa izmēru, sanāk ka aģents pārklāj vairāk nekā vienu rūtiņu, kas iet pretrunā ar fundamentālo algoritma darbības principu.

Mūsdienīgie mājas tīrīšanas roboti izmanto heiristiskas pieejas ceļa plānošanai. Ceļi, kas tiek radīti tādā veidā, nav optimāli, un nenodrošina absolūti pilnu pārklājumu, kā arī īsāko tīrīšanas laiku. Visbiežāk robotu uzvedība ir sekojoša: pēc palaišanas robots lokalizējās kartē, un apceļo tīrāmas zonas perimetru. Apceļojot perimetru robots “aizpilda” iekšējo vides platību “S” burta secībā, vai pa spirāli.

* 1. Vides dekompozīcija

Pilna pārklājuma ceļa meklēšana pēc būtības ir ceļojošā pārdevēja problēmas speciālais gadījums, kur visi mezgli ir vienādā attālumā. Tas būtiski samazina problēmas risināšanas laika sarežģītību (no O(n!) līdz O(n2)), tomēr laika sarežģītība joprojām pieaug eksponenciāli, palielinoties vides platībai. Lai panāktu algoritma optimizāciju, visbiežāk tiek pielietota vides dekompozīcija. Sadalot vidi vairākos mazākos gabalos, var samazināt punktu skaitu ko, algoritmam ir jāapskata, bet algoritms būs jāpielieto atsevišķi katram izveidotam apgabalam. Tāda metode joprojām uzlabo algoritma veiktspēju, tomēr dažādos gadījumos dekompozīcija var negatīvi ietekmēt kopējā ceļa efektivitāti. Piemēram ja sadalīt tīrāmo vidi pārāk lielā mazu apgabalu skaitā, aģents tērēs daudz laika ceļošanai starp apgabaliem, pārklājot jau izbrauktas zonas, tērējot laiku un enerģiju bezjēdzīgai braukšanai. Lai uzlabotu veiktspēju, būtiski nedraudējot efektivitāti ir jāizvēlas optimālais risinājums vides dekompozīcijai. Pārklājuma ceļa meklēšanas nozare visbiežāk pielieto Voronova diagrammu metodi (*Voronoi diagramm*), Delaunāja triangulāciju (*Delaunay triangulation*), vai trapezoidālo sadalīšanu.