Model razmještaja osjetila za pokrivanje zatvorenoga prostora

Sušanj, Diego

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET

Diego Sušanj

MODEL RAZMJEŠTAJA OSJETILA ZA POKRIVANJE ZATVORENOGA PROSTORA

DOKTORSKA DISERTACIJA

Rijeka 2021.

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET

Diego Sušanj

MODEL RAZMJEŠTAJA OSJETILA ZA POKRIVANJE ZATVORENOGA PROSTORA

DOKTORSKA DISERTACIJA

Mentor: izv. prof. dr. sc. Kristijan Lenac

Rijeka 2021.

UNIVERSITY OF RIJEKA
FACULTY OF ENGINEERING

Diego Sušanj

SPATIAL SENSOR DISTRIBUTION MODEL FOR INDOOR ENVIRONMENT COVERAGE

DOCTORAL THESIS

Rijeka 2021.

Mentor doktorske disertacije: izv. prof. dr. sc. Kristijan Lenac (Sveučilište u Rijeci, Tehnički fakultet)

Doktorski rad obranjen je dana _____ na Tehničkom fakultetu Sveučilišta u Rijeci, pred povjerenstvom u sastavu:

1. prof. dr. sc. Ivo Ipšić - predsjednik (Sveučilište u Rijeci, Tehnički fakultet)
2. doc. dr. sc. Sandi Ljubić - član (Sveučilište u Rijeci, Tehnički fakultet)
3. prof. dr. sc. Renato Filjar - član (Veleučilište Hrvatsko zagorje Krapina)

ZAHVALE

Iskrenu zahvalu upućujem mentoru, izv. prof. dr. sc. Kristijanu Lencu na savjetima, strpljenju i vođenju kroz studij i izradu ove doktorske disertacije.

Hvala Domagoju na pomoći, idejama i suradnji za ovo istraživanje, ali i na svim kavama koje smo popili.

Hvala svim kolegicama i kolegama na podršci i zajednički provedenom vremenu. Hvala Damiru i Draženu, na svim savjetima i mudrostima, žao mi je što više ne radimo skupa.

Hvala Žigi, Peteru i cijelom LRV-u, vrijeme provedeno s vama u Ljubljani mi je vratilo motivaciju za znanost.

Hvala svim prijateljima koji su stajali uz mene ovih godina.

Mama i tata, vama posebno hvala na bezuvjetnoj podršci svih ovih godina, bez vas, ovo ne bi bilo moguće. Hvala Marinu i Bartu, vi mi usrećite svaki dan.

SAŽETAK

Mnoge primjene zahtijevaju postavljanje mjernih osjetila u prostoru. Primjerice, postavljanjem kamera u muzeje omogućuje se nadzor umjetnina i praćenje osoba u prostoru muzeja. U tom slučaju potrebno je odrediti pozicije i orijentacije kamera, a jedan od važnijih kriterija odabira često je pokrivenost prostora osjetilnim sposobnostima mjernih osjetila.

Ova disertacija razmatra problem razmještaja mjernih osjetila u prostoru s ciljem što veće pokrivenosti zatvorenog prostora osjetilnim sposobnostima mjernih osjetila. Kako bi računalni program bio u mogućnosti odabrati i predložiti razmještaj osjetila potrebno je modelirati promatrani prostor i korištena mjerna osjetila. U disertaciji je predložen model prostora sa sposobnošću procjene pokrivenosti te modeli osjetilnih sposobnosti svesmjernih i usmjerenih osjetila. Predložen je i model razmještaja osjetila u zatvorenom prostoru združivanjem predloženih modela prostora i modela osjetilnih sposobnosti osjetila.

Provedeno je eksperimentalno istraživanje razmještanja različitog broja svesmjernih i usmjerenih osjetila u dvodimenzionalnim i trodimenzionalnim modelima prostora. Odabrano je i predloženo šest reprezentativnih optimizacijskih algoritama koji su uspoređeni s iscrpnih pretraživanjem. Ispitani stohastički algoritmi ostvarili su značajno ubrzanje vremena izvođenja uz malu relativnu pogrešku pokrivenosti prostora u odnosu na pristup iscrpnog pretraživanja. Obilježja šest odabranih optimizacijskih algoritama analizirana su i međusobno uspoređena za problem razmještaja osjetila u zatvorenom prostoru. Eksperimentalna provjera pokazala je da algoritam umjetnog roja pčela ostvaruje najveću prosječnu pokrivenost prostora bez značajne razlike u vremenu izvođenja odabranih algoritama.

Ključne riječi: razmještaj osjetila, model prostora, model osjetilnih sposobnosti osjetila

ABSTRACT

Many applications require the placement of sensors in an environment. For example, the placement of cameras in museums enables the monitoring of artworks and people in the museum space. In this case, it is necessary to determine the positions and orientations of the cameras. When determining sensor placement, one of the more important selection criteria is often the coverage of the environment by the sensors.

This dissertation investigates the sensor placement problem with the goal of maximizing the sensor coverage of an indoor environment. In order for the computer program to be able to select and suggest sensor placement, it is necessary to model the observed environment and the sensors used. In this dissertation, an environment model with coverage estimation capability and models of sensing abilities for isotropic and directional sensors are proposed. The model of spatial sensor distribution in an indoor environment combining the proposed environment and sensor models is proposed.

An experimental study on the spatial distribution of different number of isotropic and directional sensors is conducted. The sensors are placed in two-dimensional and three-dimensional models of the environment. Six selected optimization algorithms are compared with exhaustive search. Compared to the exhaustive search approach, the selected stochastic algorithms achieve significantly lower computation times with a small relative error of the environment coverage. The properties of the six selected optimization algorithms are studied and compared for the indoor sensor placement problem. Experimental verification shows that among the six selected algorithms, the Artificial Bee Colony algorithm achieves the highest average coverage for all test cases, with no significant differences in the execution time of the selected algorithms.

Keywords: spatial sensor distribution, environment model, sensor sensing ability model

SADRŽAJ

1. Poglavlje

UVOD

U današnje doba ubrzanog razvoja tehnologije i ljudskog napretka u korištenju iste, sveprisutno računarstvo (eng. Ubiquitous Computing) zauzima rastući ugled u društvu. Sveprisutno računarstvo predstavlja koncept u kojem je obrada informacija integrirana u predmete i aktivnosti s kojima se svakodnevno susrećemo. Primjeri tih, popularno nazvanih „pametnih", uređaja mogu se pronaći na svakom koraku, od malih kućanskih aparata, klimatizacijskih uređaja, pametnih klupa, mobitela i pločnih računala (eng. tablet), stolnih i prijenosnih računala pa do većih i složenijih industrijskih strojeva. Najveći udio navedenih uređaja za zadaću ima prikupljanje stanja okoline ili upravljanje uređajima u svojoj okolini. Sustav takvih, komunikacijski i računalno sposobnih, uređaja povezanih posebnim komunikacijskim protokolima putem Interneta naziva se Internet stvari (eng. Internet of Things). Takve uređaje nazivamo čvorovima odnosno osjetilnim čvorovima ako im je osnovna zadaća motrenje prostora u kojem se nalaze.

Osjetilni čvorovi, uz jedno ili više mjernih osjetila kojima autonomno motre prostor u kojem se nalaze, u pravilu sadrže i osnovni sustav za analizu, pohranu i slanje opažanja te izvor energije [1, 2]. Mjerna osjetila su uređaji ili dijelovi uređaja čija je svrha detektirati promjene stanja u prostoru u kojem se nalaze te poslati detektirano stanje sustavu za analizu.

1.1. Mreže osjetila

Često jedan osjetilni čvor ne može samostalno izvršiti traženu zadaću već je potrebna suradnja više osjetila u sakupljanju, razmjeni i analizi podataka. Razlog tome je potreba za sakupljanjem podataka na većem ili fizički odvojenom prostoru. Suradnja prostorno raspoređenog skupa osjetilnih čvorova naziva se mrežom osjetila. Osjetila u mreži učestalo su iste vrste, odnosno mreže su homogene, ali moguća je kombinacija više vrsta osjetila. Mreže osjetila u današnje se vrijeme uglavnom ne oslanjaju na postojeću žičnu električnu i komunikacijsku infrastrukturu već uobičajeno koriste vlastitu bežičnu mrežu za komunikaciju. U tom slučaju govorimo o podskupu mreža osjetila, bežičnim mrežama osjetila (eng. Wireless Sensor Network), decentraliziranom obliku mreže osjetila inicijalno razvijenom u istraživačkim uredima vojnih industrija s ciljem nadzora bojišta. Današnja upotreba bežičnih mreža osjetila pokriva niz industrijskih i korisničkih primjena [3, 4].

Mreže osjetila nalaze svoju osnovnu primjenu u nadzoru prostora. Osjetila se raspoređuju na prostoru kojeg je potrebno nadzirati, primjerice oko naftovoda ili plinovoda. Mreže osjetila se koriste i u zatvorenim prostorima, primjerice osjetila se raspoređuju na ključnim prolazima. Čest primjer je i nadzor pristupa muzejima i galerijama kod kojih se mogu dodatno nadzirati i prostori oko izložbenih primjeraka. Mreže osjetila, a posebice podskup bežičnih mreža, pronalaze svoju primjenu i u medicini. Bežično osjetilo se ili nosi, kao primjerice ogrlica i slično, ili se ugrađuje kao implantat u ljudsko tijelo. Postoji cijeli niz mogućih primjena, ali prvenstveno se koriste za dohvaćanje podataka o zdravlju ili fizičkoj spremi pojedinca [5, 6, 7]. Jedna od primjena bežičnih mreža osjetila je i zaštita okoliša. Primjeri uključuju praćenje zagađenja zraka, rano otkrivanje šumskih požara i klizišta, ranu detekciju indikatora prirodnih katastrofa, praćenje kvalitete vode i detekciju kemijskih sredstava, primjene u poljoprivredi i slično [8]. U industriji se bežične mreže osjetila mogu koristiti za nadzor industrijskih procesa, bez potrebe za dodatnom električnom i komunikacijskom infrastrukturom [9, 10].

Nekoliko je osnovnih problema koje je potrebno istražiti i riješiti prije korištenja bežičnih mreža osjetila [11, 12, 13, 14, 1, 15, 16, 2]. Većina problema vezana je uz razmještaj osjetila u prostoru: postizanje željene pokrivenosti prostora (eng. coverage), povezanosti (eng. connectivity) i energetske učinkovitosti (eng. energy efficiency).

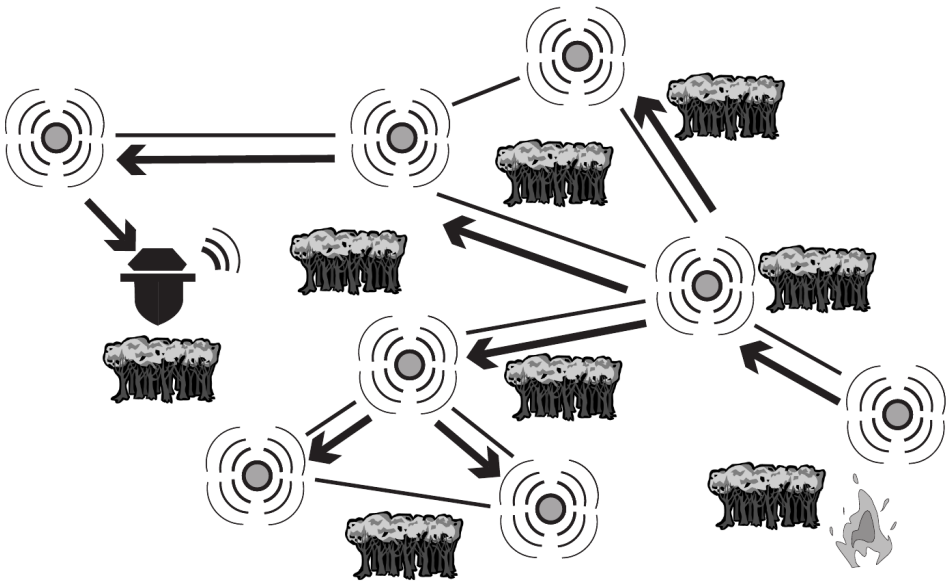
Pokrivenost promatranog prostora u osnovi predstavlja mjeru kvalitete usluge (eng.

quality of service) mreže osjetila [11]. Pokrivenost promatranog prostora ovisi o broju osjetila te o njihovom razmještaju u prostoru. Razmještaj osjetila predstavlja odabir pozicija i, kada je to potrebno, orijentacija osjetila.

Povezanost mreže utječe na sposobnost da se sakupljeni podatci iz prostora proslijede do sustava za obradu podataka. Za tu je primjenu važno razmotriti i otpornost mreže na utjecaj pogrešaka, primjerice kvara na nekom od osjetila.

Energetska učinkovitost kod bežičnih mreža osjetila je izrazito bitna značajka. Osjetila su uglavnom opremljena baterijama, a ponekad i sustavima za dobivanje energije iz okoline (eng. energy harvesting). Kako su osjetila često raspoređena u prostorima s opasnim uvjetima za čovjeka, zamjena baterija je otežana te je pri razvoju i korištenju sustava potrebno voditi računa o potrošnji energije.

Razmještaj (eng. deployment) osjetila moguć je na dva načina: nasumice ili planirano. Različiti problemi proizlaze iz svakog od načina razmještaja osjetila te se razlikuju, ovisno o razmještaju, algoritmi i metode analize za ranije navedene probleme. Primjer nasumičnog razmještaja osjetila je bacanje većeg broja osjetila iz aviona nad širim prostorom kojeg se želi nadzirati, primjerice s ciljem rane detekcije požara, kako je prikazano na slici 1.1. Planski razmještaj osjetila čest je u sigurnosnim primjenama i prije samog razmještaja je potrebno odrediti pozicije samih osjetila.



Slika 1.1: Primjer bežične mreže osjetila za ranu detekciju požara [17]

Predmet ovog istraživanja je planirani razmještaj osjetila u zadanom, poznatom,

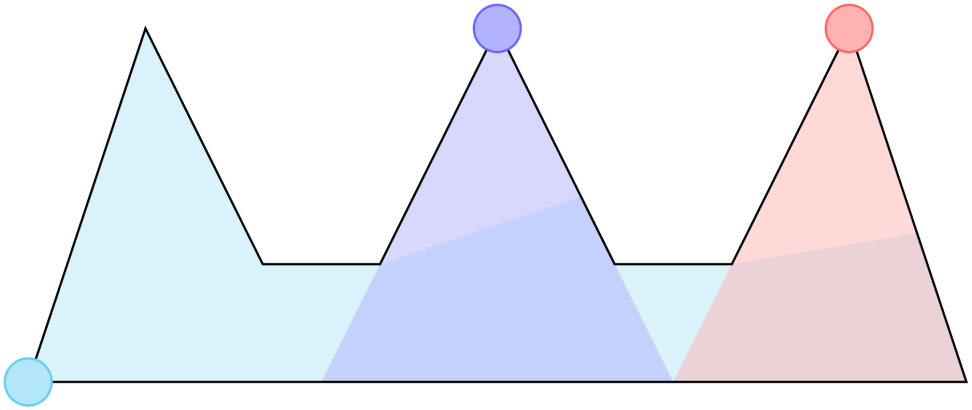
zatvorenom prostoru. Povezanost i energetska učinkovitost nisu obuhvaćene ovim istraživanjem, već se razmatra samo problem pokrivenosti prostora prethodno poznatim brojem osjetila. Scenariji koji ovise o pokrivenosti prostora uglavnom su vezani uz sigurnosne primjene, ali mogu se pronaći i kod niza primjena praćenja i analize ponašanja osoba. U sigurnosnim primjenama kamere ili druga osjetila najčešće pokrivaju specifične dijelove prostora većeg značaja, primjerice prolaze, prostor oko izloženih umjetnina i slično. Kod analize ponašanja osoba, povećana pokrivenost prostora ispred polica s proizvodima, promocijskih plakata ili izloga, omogućuje prikupljanje opažanja za potrebe analize zainteresiranosti osoba za određeni proizvod ili vrstu reklame te samim time modeliranje pozicija proizvoda ili vrste promidžbe [16, 18, 19, 20].

1.2. Problem umjetničke galerije

Problem razmještaja osjetila s ciljem rješavanja problema pokrivenosti prostora pronalazi svoju osnovu u poznatom problemu računalne geometrije, problemu umjetničke galerije. Osnovu problema umjetničke galerije postavio je Victor Klee 1973. godine [21] kroz pitanje: *„Koji je najmanji broj čuvara potreban za čuvanje umjetničke galerije?"*. Ovo pitanje u danom je trenutku predstavljalo zanimljiv geometrijski problem te je Vasek Chvátal prezentirao svoja prva opažanja 1975. godine [22].

Model prostora umjetničke galerije opisuje se konačnim (eng. finite) i zatvorenim (eng. closed) poligonom sastavljenim od vrhova i bridova. U većini varijacija problema, umjetnička galerija opisana je jednostavnim poligonom (eng. simple polygon), odnosno kao prostor na ravnini unutar poligona koji se ne presijeca [23]. Čuvari (eng. guards) se ovisno o poziciji na kojoj se nalaze ili mogu nalaziti, dijele na čuvare vrha (eng. vertex guard), čuvare točke (eng. point guard) i čuvare brida (eng. edge guard). Čuvar vrha može se nalaziti samo u vrhovima poligona u ravnini koji opisuje umjetničku galeriju, čuvar brida se može nalaziti samo na bridu poligona, dok su moguće pozicije čuvara točke sve točke unutar prostora omeđenog poligonom u ravnini. Dodatno, u izvedenicama problema umjetničke galerije, definira se i pokretljivi čuvar (eng. mobility guard, *watchman*) kojemu je dozvoljeno kretanje po slijedu ravnih segmenata potpuno sadržanih unutar poligona [24]. Primjer jednostavne umjetničke galerije s tri čuvara vrha prikazan je na slici 1.2.

Kleeov originalni problem umjetničke galerije pretpostavlja sposobnost čuvara da mo-



Slika 1.2: Primjer prostora jednostavne umjetničke galerije s razmještena tri čuvara vrha [20]

tre prostor bez ikakvih ograničenja, kako po pitanju udaljenosti, tako i kutu oko njihove pozicije, odnosno mogućnost motrenja od 360° oko statične pozicije čuvara. Ovakva definicija čuvara pretpostavlja samo optičku vidljivost (eng. line-of-sight), odnosno binarnu vidljivost gdje su dvije točke međusobno vidljive ako segment linije između njih ne presijeca objekt u prostoru ili sam poligon u ravnini. Kasnija istraživanja dodala su ograničenja vidljivosti, primjerice udaljenosti i ograničenog vidnog polja (eng. field of view) [25].

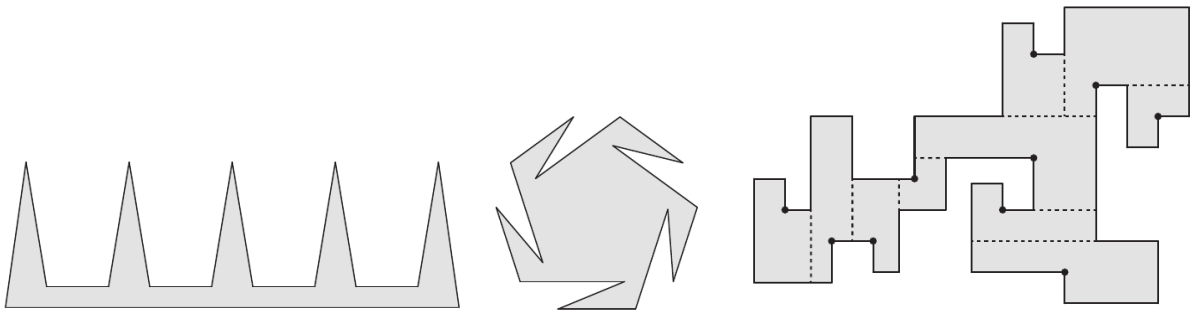
Chvátal je pokazao da je za poligone s n vrhova povremeno potrebno i uvijek dovoljno $\lfloor n/3 \rfloor$ čuvara [22]. Fisk [26] 1978. godine predlaže jednostavniji dokaz koristeći postupke triangulacije i „bojanja" vrhova poligona. Desetak godina kasnije, 1987. godine, O'Rourke daje pregled dotadašnjih istraživanja i varijacija problema umjetničke galerije [27]. Nakon O'Rourkea objavljeno je nekoliko kvalitetnih preglednih radova temeljenih na problemu umjetničke galerije. Shermer je 1992. godine objavio rad [28] u kojem daje pregled rješenja za niz poligona posebnih oblika. U [29] autor Jorge Urrutia daje pregled i opis problema umjetničke galerije na jednostavan način razumljiv široj publici.

Dosad navedena istraživanja koristila su dvodimenzionalni model umjetničke galerije, odnosno, model opisan poligonom u ravnini. Ovakav model opisuje tlocrt umjetničke galerije te ne pruža uvijek dovoljno informacija o složenim prostornim strukturama. Trodimenzionalni modeli prostora omogućuju vjerniji opis promatranog prostora, odnosno opis koji uključuje trodimenzionalne prostorne značajke poput visina i oblika prepreka koje se nalaze u prostoru. Vjerniji opis promatranog prostora neophodan je pri odlučivanju o razmještaju čuvara u promatranom prostoru [30]. U dosadašnjim istraživanjima trodimenzionalni modeli umjetničke galerije su uglavnom opisivani visinskim kartama, dvodimenzi-

onalnim matricama s vrijednostima visine prostora u svakoj točki [31] ili na ortogonalnim poliedarskim (eng. orthogonal polyhedra) modelima [32].

Rješenja problema umjetničke galerije, za dvodimenzionalne i trodimenzionalne modele prostora, spadaju u razred NP-teških problema [33, 34], ali postoje matematički dokazana rješenja za neke dvodimenzionalne i trodimenzionalne prostore opisane jednostavnim poligonom [27, 29, 25, 35].

Na slici 1.3 dani su primjeri dvodimenzionalnih modela umjetničke galerije poznatih rješenja. Na slici 1.3 a) prikazan je poligon „češlja" (eng. comb polygon) s k vrhova „češlja" odnosno s $3k$ točaka poligona. Za pokriti navedeni primjer umjetničke galerije potrebno je k čuvara razmještenih u vrhove „češlja". Na slici 1.3 b) dan je primjer umjetničke galerije poligona „zatvarača" (eng. shutter polygon). U navedenom primjeru barem po jedan čuvar treba biti razmješten za svaki konveksni dio poligona. U prikazanom primjeru potrebno je pet čuvara. Na slici 1.3 c) dan je prikaz ortogonalnog dvodimenzionalnog poligona (eng. orthogonal polygon) podijeljenog u elemente „L" oblika. Za primjer sa slike c) je potrebno razmjestiti 10 čuvara.

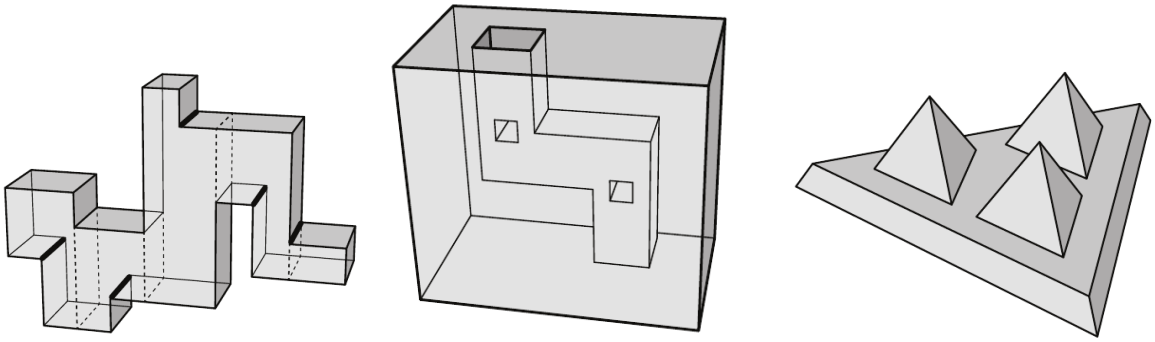


(a) (b) (c)

Slika 1.3: Primjeri dvodimenzionalnih modela umjetničke galerije s poznatim rješenjima [32]

Na slici 1.4 dani su primjeri raznih trodimenzionalnih modela umjetničke galerije. Na slici 1.4 a) dan je primjer monotonog ortogonalnog poliedra. Kao što je to bilo u primjeru na slici 1.3 c) i ovdje je model prostor podijeljen u elemente „L" oblika. Svaki od tih „L" elemenata zahtijeva jednog čuvara brida. U prikazanom primjeru potrebno je razmjestiti pet čuvara brida kako bi prostor bio u potpunosti pokriven. Na slici 1.4 b) dan je primjer ortogonalnog polieadra s prazninom unutar prostora. Pri određivanju broja čuvara za poliedarske modele s prazninama moguće je odrediti samo gornju granicu koja ovisi o

broju praznina i njihovoj povezanosti. Na slici 1.4 c) prikazan je posebni slučaj poliedra za koji je potrebno tri čuvara brida.



(a) (b) (c)

Slika 1.4: Primjeri trodimenzionalnih modela umjetničke galerije s poznatim rješenjima [32]

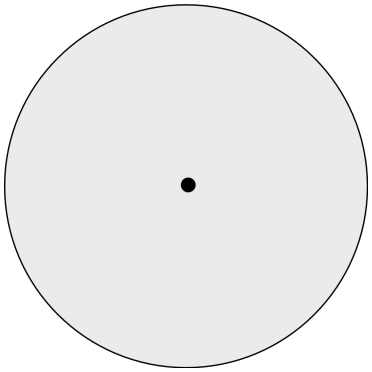
Stvarne primjene rješenja temeljenih na problemu umjetničke galerije uključuju razmještanje zaštitara, sigurnosnih kamera, svjetlosnih instalacija ili radarskih stanica [36, 37]. Primjene koje proizlaze iz problema umjetničke galerije u načelu su složenije od samog problema umjetničke galerije zbog geometrijske složenosti i raznolikosti oblika koji se pronalaze u stvarnom svijetu te upravo zbog toga pronalazak općenitog rješenja pogodnog za sve oblike poligona umjetničke galerije nije moguć [30].

Na složenost rješenja praktičnih realizacija problema umjetničke galerije utječe i dimenzionalnost prostora u kojem je potrebno izvršiti određivanje razmještaja osjetila. Izračun pokrivenosti jednostavniji je za dvodimenzionalne modele prostora, međutim opis promatranog prostora dvodimenzionalnim modelom može dovesti do razlike u pokrivenosti koju određeni razmještaj ostvari u dvodimenzionalnom modelu prostora i pokrivenosti koju bi taj razmještaj ostvario u promatranom trodimenzionalnom prostoru. Trodimenzionalni modeli prostora daju vjerniji opis promatranog prostora i uobičajeno manju pogrešku u procjeni pokrivenosti prostora, ali s problemom većeg računskog troška zbog dodatne dimenzije koja se modelira. Pri modeliranju prostora potrebno je uzeti u obzir i topografiju prostora. Topografija prostora uključuje oblik prostora te postojeće prepreke koje prekrivaju osjetilno područje svakog osjetila [38].

1.3. Osjetilne sposobnosti osjetila

Osjetila su opisana svojom osjetilnom sposobnosti. Osjetilna sposobnost osjetila u osnovi ovisi o dvije usko povezane značajke, svom obliku i vrsti, vrijednosti koju može poprimiti.

Ovisno o obliku osjetilnog područja osjetila dijelimo na svesmjerna i usmjerena. Primjeri često korištenih modela osjetilnih područja u dosadašnjim istraživanjima dani su na slikama 1.5 i 1.6 [39]. Osjetilno područje osjetila svesmjerne osjetilne sposobnosti prostire se u svim smjerovima od samog osjetila, neovisno o kutu. Zbog oblika osjetilnog područja, svesmjerna osjetila često se nazivaju diskovima. Na slici 1.5 prikazan je model svesmjernog osjetilnog područja.

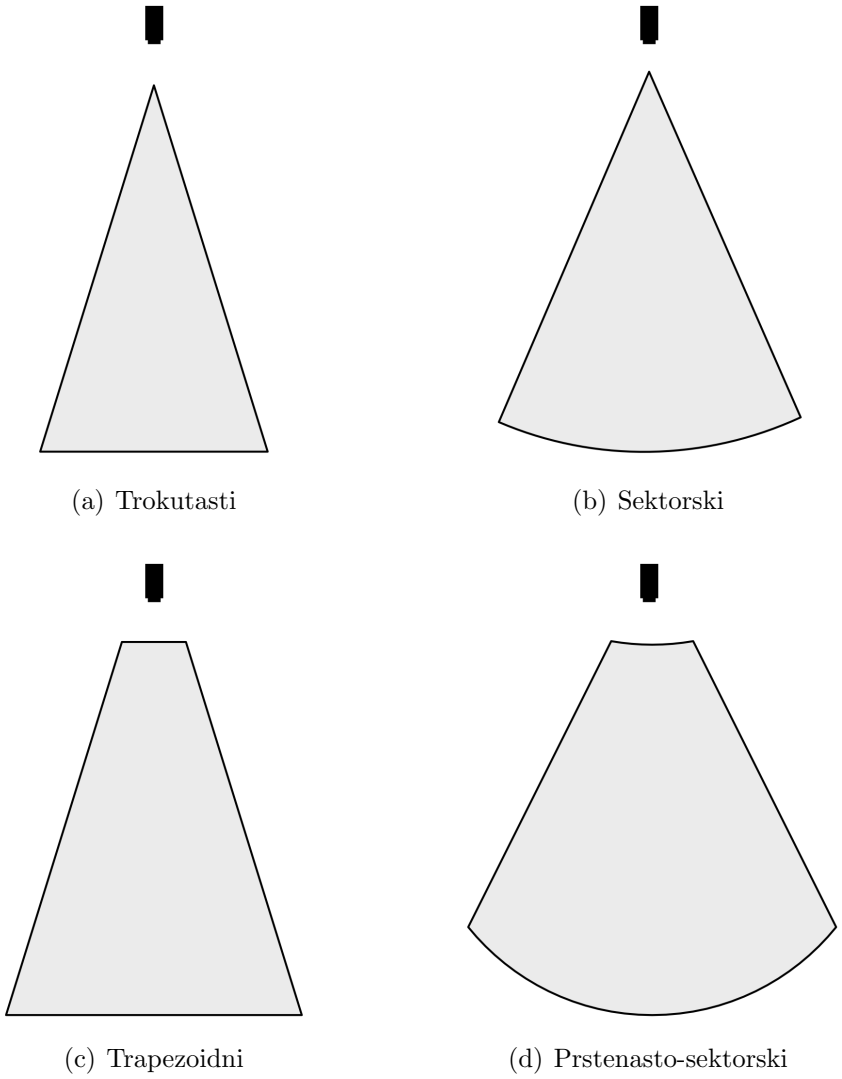


Slika 1.5: Model dvodimenzionalnog svesmjernog osjetilnog područja

Za osjetila usmjerene osjetilne sposobnosti osjetilno područje ovisi i o kutu pravca koji prolazi kroz točku kojom je definirana pozicija osjetila u Kartezijevom koordinatnom sustavu te kroz bilo koju točku u prostoru. Dosadašnja istraživanja uglavnom su koristila dvodimenzionalne osjetilne sposobnosti, ali se te osjetilne sposobnosti mogu preslikati i u trodimenzionalne.

Na slikama 1.6 a) i b) dani su primjeri modela usmjerenog osjetilnog područja s početkom u točki osjetila. Na slikama 1.6 c) i d) dani su modeli slični onima iz a) i b), ali s odmakom početka osjetilnog područja od osjetila. Ovo je čest slučaj kod raznih usmjerenih osjetila, primjerice kamera, kod kojih nije moguće detektirati ili raspoznati objekte koji se nalaze u neposrednoj blizini osjetila.

Uz oblik osjetilnog područja bitna je i vrsta osjetilne sposobnosti iskazana kroz vrijednost vidljivosti svake točke prostora. Vidljivost je mjera koja za svaku točku prostora



(a) Trokutasti (b) Sektorski

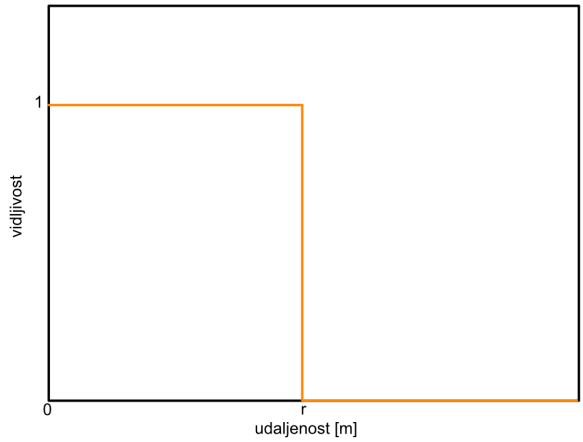
(c) Trapezoidni (d) Prstenasto-sektorski

Slika 1.6: Modeli dvodimenzionalnih usmjerenih osjetilnih područja

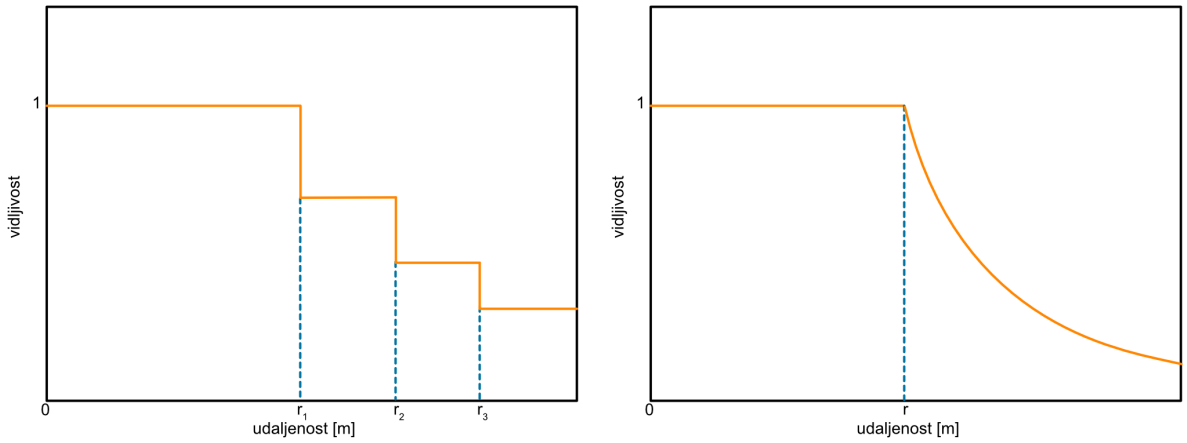
opisuje je li vidljiva, i koliko dobro, od strane osjetila. Primjerice, vidljivost može opisivati sposobnost osjetila da detektira objekte unutar svog osjetilnog područja ili da detektira neki elektromagnetski signal u prostoru. Osjetila se obično modeliraju korištenjem binarne ili vjerojatnosne osjetilne sposobnosti.

Binarni model pokrivanja opisuje vidljivost u samo dvije razine, nula za točke prostora koje su izvan te jedan za točke koje su unutar osjetilnog područja osjetila. Na slici 1.7 dan je primjer modela binarne osjetilne sposobnosti s dometom potpuno vidljivog prostora do zadane udaljenosti r. Prostor i objekti u prostoru udaljeni više od zadane udaljenosti r nisu vidljivi.

Za razliku od binarnog, vjerojatnosni model pokrivanja opisuje vidljivost svake točke

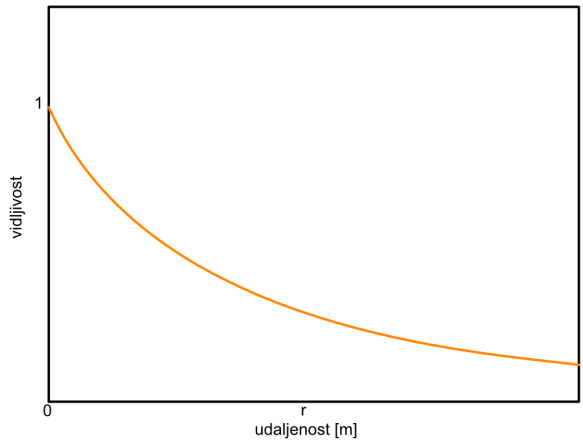


Slika 1.7: Model binarne osjetilne sposobnosti [40]



(a) Stepenasti

(b) Stepenasto-eksponencijalni



(c) Eksponencijalni

Slika 1.8: Modeli vjerojatnosne osjetilne sposobnosti [40]

prostora s vrijednošću između nula i jedan te omogućuje bolji opis vidljivog prostora u okolini osjetila. Korištenjem binarnog modela pokrivanja nije moguće modelirati situacije

djelomične vidljivosti, odnosno uvodi se pogreška pri pretpostavci o potpunoj vidljivosti dijela prostora ili nevidljivosti prostora koji je djelomično vidljiv.

Na slici 1.8 a) prikazan je stepenasti model kod kojeg je osjetilo opisano vjerojatnosnom osjetilnom sposobnošću, ali s diskretnim razinama vidljivosti ovisno o udaljenosti [41]. Na slikama 1.8 b) i c) dani su primjeri modela vjerojatnosne osjetilne sposobnosti proizašle iz eksponencijalnih funkcija. Model prikazan na slici 1.8 b) pretpostavlja potpunu vidljivost do zadane udaljenosti r te potom vidljivost opada s udaljenosti [42, 43]. Kod modela prikazanog na slici 1.8 c) potpuna je vidljivost samo u točki osjetila te potom opada s udaljenosti [44].

1.4. Metrike pokrivenosti prostora

Uz dimenzionalnost i topografiju prostora te osjetilna područja i sposobnosti pojedinačnih osjetila u postupku razmještaja osjetila bitan je i pristup vrednovanju pokrivenosti prostora. Metrike pokrivenosti prostora predstavljaju definicije specifičnih kriterija koji se mogu koristiti pri izračunu pokrivenosti zadanog područja ili točaka u prostoru [2, 45, 46, 47, 48]. Odabir metrike pokrivenosti ovisi o problemu koji se rješava te cilju razmještaja osjetila [4, 49, 11].

Pri određivanju razmještaja osjetila, za izračun pokrivenosti prostora osjetilnim područjima osjetila, prostor se obično podijeli na konačan broj elemenata, ispitnih točaka prostora. Ti elementi se kod dvodimenzionalnih prostora nazivaju pikseli (eng. pixel) i obično su kvadratnog oblika, odnosno sve četiri stranice su im jednake veličine. Kod trodimenzionalnih prostora najmanji elementi prostora nazivaju se vokseli (eng. voxel) i obično imaju oblik kocke, odnosno sve su im stranice jednake veličine.

Često korištena metrika pokrivenosti prostora je pokrivenost područja (eng. area coverage). Ova metrika predstavlja omjer prostora pokrivenog osjetilima naspram cijelog prostora [14, 50]. Kod binarnih osjetilnih sposobnosti ta metrika je izražena kao omjer broja točaka prostora unutar osjetilnih područja osjetila v s ukupnim brojem točaka prostora n:

$$C = \frac{v}{n}. \tag{1.1}$$

Kod vjerojatnosnih osjetilnih sposobnosti, metrika pokrivenosti područja ne uzima u

obzir samo omjer broja točaka prostora unutar osjetilnih područja osjetila već i vrijednost njihove vidljivosti [40]:

$$C = \frac{1}{n} \sum_{i=1}^{n} v_i. \tag{1.2}$$

Vidljivost v_i svake točke u prostoru računa se ovisno o razmještaju osjetila u prostoru te njihovim osjetilnim sposobnostima.

Moguće je dijelovima prostora dati veći značaj od ostatka. Tada je, uz vidljivost v_i svake točke prostora, potrebno definirati i značaj svake točke koristeći težinski faktor w_i:

$$C = \frac{1}{\sum_{i=1}^{n} w_i} \sum_{i=1}^{n} (w_i \cdot v_i). \tag{1.3}$$

Metrike pokrivenosti prostora koje se zasnivaju na korištenju težina koje daju veći značaj dijelu prostora su metrika pokrivenosti točke (eng. point coverage) i metrika pokrivenosti prepreke (eng. barrier coverage) [51].

Koristi se još i k-pokrivenost (eng. k-coverage) čiji je cilj pokriti svaku točku prostora s najmanje k osjetila. Svaka točka koja nije vidljiva iz barem k osjetila ima vidljivost nula. Prethodno opisane formule mogu se i ovdje koristiti, uz proširenje uvjetom o vidljivosti iz barem k osjetila. Neki od razloga primjene k-pokrivenosti pri razmještaju osjetila su povećanje robusnosti i omogućavanje lokalizacije, određivanja položaja, objekata u u nekoj točki prostora [52]. Robusnost se osigurava pokrivanjem svake točke prostora s većim brojem osjetila, kako bi u slučaju prestanka rada jednog od osjetila točka prostora i dalje bila vidljiva od barem jednog osjetila. Lokalizacija objekata koji se nalaze u prostoru, omogućava se pokrivanjem te točke s više osjetila te uz nekoliko preduvjeta poput broja osjetila i njihovog međusobnog odnosa. Broj k potrebnih osjetila ovisi o dimenzionalnosti prostora te o pristupu korištenom pri lokalizaciji.

Sve prethodne metrike pretpostavljaju da se prostor motri neprekidno. Metrika pregleda područja (eng. sweep coverage) opisuje mogućnost da se svaka točka prostora nadzire sporadično, što se postiže korištenjem mobilnih osjetila.

1.5. Hipoteza i očekivani znanstveni doprinosi istraživanja

Dosadašnja istraživanja, opisana u poglavlju 2., promatrani prostor su opisivali dvodimenzionalnim modelima ili visinskim kartama prostora. Autor ovog rada uočio je nedostatak rješenja koji promatrani prostor opisuju trodimenzionalnim modelima prostora. Problem većeg računskog troška koji nastaje korištenjem trodimenzionalnih modela prostora moguće je olakšati koristeći modele prostora koji omogućuju učinkovit, računski jednostavni, izračun vidljivosti i procjenu pokrivenosti prostora.

U dosadašnjim istraživanjima osjetila su opisivana koristeći binarnu ili vjerojatnosnu osjetilnu sposobnost. Pritom je vjerojatnosna osjetilna sposobnost opisivana jednostavnim funkcijama, koje pojednostavljeno opisuju stvarnu osjetilnu sposobnost osjetila.

Temeljem uočenih nedostataka reprezentativnih modela trodimenzionalnih zatvorenih prostora i reprezentativnih modela osjetilnih sposobnosti osjetila definirana je hipoteza istraživanja:
Postupkom združivanja modela prostora i osjetilne sposobnosti osjetila moguće je kreirati model razmještaja osjetila u zatvorenom prostoru.

Temeljem navedenog, definirani su i znanstveni doprinosi ovog istraživanja:

1. prijedlog modela prostora sa sposobnošću procjene pokrivenosti prostora;
2. prijedlog modela osjetilne sposobnosti predstavnika svesmjernih i usmjerenih osjetila;
3. prijedlog modela za razmještaj osjetila u zatvorenom prostoru združivanjem modela prostora i osjetila.

1.6. Metodologija istraživanja

Istraživanje je provedeno u četiri faze usko vezane uz ostvarivanje znanstvenih doprinosa ovog istraživanja:

– Cilj prve faze bila je izrada modela prostora koji omogućuj procjenu njegove pokrivenosti. Rezultat ove faze takav je generički model prostora koji omogućava izradu

skupa dvodimenzionalnih i trodimenzionalnih modela prostora. Kroz ovu fazu ostvaren je prvi znanstveni doprinos.

– U drugoj fazi cilj je bio proučiti osjetilne sposobnosti više vrsta osjetila te ih kategorizirati ovisno o tome odašilju li ili primaju signal, imaju li usmjerenu ili svesmjernu osjetilnu sposobnost te modelirati primljeni, odnosno odaslani, signal ovisno o udaljenosti i kutu od osjetila. Rezultat ove faze su generički modeli osjetila opisani kroz funkcije pokrivenosti područja ovisno o udaljenosti i kutu između točke u prostoru i samog osjetila. Temeljem navedenih generičkih modela osjetila izvedeni su i modeli svesmjernog te usmjerenog osjetila koji realistično opisuju osjetilnu sposobnost izabranih osjetila. U ovoj fazi ostvaren je drugi znanstveni doprinos.

– Kroz treću fazu proučene su postojeće metrike pokrivenosti prostora. Kako je cilj istraživanja odrediti razmještaj osjetila za pokrivanje cijelog promatranog prostora, odabrana je metrika pokrivenosti područja. Koristeći odabranu metriku pokrivenosti područja te prethodno predložene modele prostora i osjetila predložena je optimizacijska funkcija. Cilj ove faze bilo je združivanje modela osjetila, modela prostora te optimizacijske funkcije temeljene na odabranoj metrici u model za razmještaj osjetila u zatvorenom prostoru. Ova je faza rezultirala ostvarivanjem trećeg znanstvenog doprinosa.

– Kroz ovo istraživanje predložen je model za razmještaj osjetila u zatvorenom prostoru združivanjem modela prostora i osjetila. Posljednja, četvrta faza, uključivala je istraživanje i odabir odgovarajućih optimizacijskih algoritama te njihovu usporedbu za rješavanje problema razmještaja osjetila. Odabrani algoritmi uspoređeni su temeljem vrijednosti pokrivenosti prostora koju su ostvarili razmještanjem te vremenom njihovog izvođenja. Izvršena je i eksperimentalna usporedba iscrpnog pretraživanja i stohastičkih optimizacijskih algoritama za problem razmještaja osjetila

1.7. Struktura doktorske disertacije

Ova doktorska disertacija organizirana je u osam poglavlja. U ovom, prvom poglavlju, definirani su osnovni pojmovi vezani uz problem razmještaja osjetila, opisan je problem umjetničke galerije, opisani su modeli osjetilnih sposobnosti osjetila i metrike pokrivenosti

prostora. U ovom poglavlju su definirani hipoteza i znanstveni doprinosi ovog istraživanja. Uz navedeno, dan je pregled metodologije istraživanja.

U drugom poglavlju je napravljen pregled i analiza dosadašnjih istraživanja vezanih uz problem razmještaja osjetila.

Treće poglavlje opisuje predložen model prostora sa sposobnošću procjene pokrivenosti prostora. Opisane su i osnovne funkcije korištene za procjenu pokrivenosti te pristup rasterizaciji prostora. U ovom je poglavlju prezentiran prvi znanstveni doprinos ovog istraživanja.

Pristup modeliranju osjetilnih sposobnosti osjetila te modeli osjetilnih sposobnosti predstavnika svesmjernih i usmjerenih osjetila predloženi su u četvrtom poglavlju te predstavljaju drugi znanstveni doprinos istraživanja.

U petom poglavlju predložena je optimizacijska funkcija temeljena na metrici pokrivenosti područja kao sastavni dio modela razmještaja osjetila u zatvorenom prostoru temeljenog na združivanju modela prostora i osjetila. U ovom je poglavlju prezentiran treći znanstveni doprinos ovog istraživanja.

Opis izrađenog simulacijskog okvira sa pripadajućim segmentima opisa prostora, osjetilnih sposobnosti osjetila te optimizacijskog zadatka dan je u šestom poglavlju.

Sedmo poglavlje sadrži opis odabranih postavki eksperimenata koji koriste model za razmještaj osjetila iz ranijeg poglavlja. U ovom su poglavlju opisani modeli prostora i osjetilnih sposobnosti osjetila te njihove postavke korištene pri eksperimentalnoj provjeri modela razmještaja osjetila u zatvorenom prostoru. Opisani su i odabrani optimizacijski algoritmi te vrijednosti njihovih značajki.

Usporedba iscrpnog pretraživanja i stohastičkih algoritama za rješavanje problema razmještaja osjetila te rezultati optimizacije razmještaja osjetila u odabranim modelima prostora dani su u osmom poglavlju. U ovom poglavlju napravljena je i analiza te usporedba odabranih optimizacijskih algoritama te je prezentirano nekoliko primjera odabranog razmještaja.

U posljednjem, devetom poglavlju, napravljen je konačni pregled rješenja i doprinosa istraživanja. Kratko su opisani zbirni rezultati usporedbe algoritama, dan je kritički osvrt te je navedeno nekoliko problema i pristupa koji predstavljaju buduća istraživanja.

2. Poglavlje

ANALIZA POSTOJEĆIH ISTRAŽIVANJA

Autori Dhillon i Chakrabarty u radu [53] povezali su razmještaj osjetila u dvodimenzionalnom prostoru s problemom umjetničke galerije. Prostor i moguće pozicije na koje se osjetila mogu razmjestiti su u tom prostoru diskretizirane, a osjetila su modelirana binarnom svesmjernom osjetilnom sposobnosti, odnosno diskovima. U radu je napravljena analiza utjecaja razmještaja i broja osjetila na pokrivenost promatranog prostora.

Hörster i Lienhart u radu [54] predlažu jednostavne modele usmjerenih osjetila, kamera, binarne osjetilne sposobnosti. Ta osjetila se razmještaju u dvodimenzionalnom prostoru koristeći postupak cjelobrojnog linearnog programiranja (eng. Integer Linear Programming) koji optimizira zadanu funkciju cilja. Isti su autori nastavili istraživanje u radu [55] uspoređujući tri pristupa, binarno cjelobrojno programiranje (eng. Binary Integer Programming), pohlepno pretraživanje (eng. Greedy Search) te nasumično razmještanje osjetila.

Bodor i suradnici prezentirali su u radu [18] primjer korištenja usmjerenih osjetila binarne osjetilne sposobnosti u dvodimenzionalnom prostoru uz određivanje njihovog optimalnog razmještaja. U obzir se uzima maksimalna pokrivenost zadanog dijela prostora pri definiranju funkcije cilja. U odnosu na rad Hörstera i Lienharta [55], model osjetila je nadograđen tako da je dio vidnog područja blizu kamere postavljen kao nevidljivi dio vidnog područja, što odgovara stvarnim osjetilima.

Pålsson i Ståhl u radu [56] predložili su rješenje za razmještaj usmjerenih osjetila

binarne osjetilne sposobnosti u dvodimenzionalni prostor opisan poligonom te ga povezali s problemom umjetničke galerije. U dijelu rada raspravljaju o potencijalnim poboljšanjima predloženog algoritma te prvi spominju dodavanje podrške za visinske karte, odnosno polu-trodimenzionalni ili 2,5-dimenzionalni problem. Ta poboljšanja doprinose vjernijem razmještaju osjetila.

Gonzalez-Barbosa sa suradnicima je u radu [57] predložio rješenje za određivanje optimalnog razmještaja svesmjernih i usmjerenih osjetila koristeći binarno cjelobrojno programiranje. Obje vrste osjetila modelirane su koristeći binarnu osjetilnu sposobnost.

Hefeeda i Ahmadi su u radu [40] opisali problematiku razmještaja osjetila bežične mreže osjetila sa svesmjernom vjerojatnosnom osjetilnom sposobnosti u dvodimenzionalnom prostoru. U radu su predložili protokol za vjerojatnosno pokriće (eng. Probabilistic Coverage Protocol) te opisali njegovu primjenu za pokrivenost područja i k-pokrivenost.

U radu [58] Salehizadeha i suradnika, prezentirana je metoda učinkovitog razmještaja skupova svesmjernih osjetila binarne osjetilne sposobnosti u dvodimenzionalnom prostoru. Osjetila su bila podijeljena na stacionarne i mobilne. Stacionarna osjetila su se trajno nalazila na zadanim pozicijama dok su se mobilna razmještala. Za razmještanje osjetila koristio se evolucijski algoritam pojedinačne optimizacije čestica (eng. Individual Particle Optimization) inspiriran prirodom i prirodnim pojavama.

Morsley i suradnici u radu [59] su prezentirali modele usmjerenih osjetila binarne osjetilne sposobnosti. Prezentirane modele prostora povezali su s osnovnim problemom umjetničke galerije te ih modelirani koristeći jednostavne poligone. Ispitali su dva pristupa za određivanje razmještaja osjetila, dvodimenzionalni pristup te razmještaj osjetila na zadanoj visini uz orijentaciju osjetila prema tlocrtu ravnine poda (eng. top-down). Autori su u drugom pristupu iskoristili projekciju usmjerenog osjetila na ravninu poda. Time efektivno zamjenjuju 2,5-dimenzionalni pristup sa svesmjernim osjetilom u dvodimenzionalnom prostoru. Na oba pristupa ispitali su nekoliko modifikacija algoritma optimizacije rojem čestica (eng. Particle Swarm Optimization), binarni genetski algoritam (eng. Binary Genetic Algorithm) te simulirano kaljenje (eng. Simulated Annealing).

Akbarzadeh i suradnici kroz nekoliko radova na temu razmještaja osjetila u prostoru postavljaju temelje trodimenzionalnih modela prostora. U radu [38] prostor je modeliran koristeći visinske karte prostora, a svesmjerna osjetila binarnih osjetilnih sposobnosti razmještaju se na zadanu visinu od jednog metra iznad tla. Ovaj pristup sličan je dvodimen-

zionalnom problemu, ali visina na kojoj se razmještaju osjetila te visina prostora utječu na vidljivo područje svakog osjetila. U ovome radu uspoređen je deterministički pristup razmještaju osjetila, neovisno o visini prostora, s pristupom temeljenim na korištenju algoritma strategije evolucije adaptacijom matrice kovarijance (eng. Covariance Matrix Adaptation Evolution Strategy) koji je uzimao u obzir i utjecaj visinskih prepreka na vidljivost pri optimizaciji. U promatranim se slučajevima evolucijska strategija pokazala boljom od determinističke. Dodatno, predložen je i usmjereni model osjetila, koji ima kut vidljivosti četvrtine punog kruga, odnosno kut od 90°.

U radu [60] autori proširuju svoje istraživanje s usmjerenim osjetilima vjerojatnosne osjetilne sposobnosti. Funkcija vidljivosti u ovisnosti o udaljenosti od osjetila odgovarala je sigmoidnoj funkciji dok je funkcija vidljivosti u ovisnosti o kutu sličila Gaussovoj funkciji. Odgovarajućom kombinacijom ovih dviju funkcija, dobiven je vjerojatnosni model osjetilne sposobnosti osjetila. U razmještaju osjetila uspoređeni su pristup determinističkog razmještaja, simulirano kaljenje, metoda ograničene memorije Broyden-Fletcher-Goldfarb-Shanno (eng. Limited-memory Broyden-Fletcher-Goldfarb-Shanno) te algoritam strategije evolucije adaptacijom matrice kovarijance. I u ovom slučaju, algoritam strategije evolucije adaptacijom matrice kovarijance dao je bolje rezultate od ostalih metoda.

Autori su u radu [61] proširili vjerojatnosni model osjetilne sposobnosti osjetila s još jednim kutem orijentacije osjetila. Uz kut zaošijanja (eng. yaw) u model je dodan i kut nagiba (eng. pitch). Oba kuta su, za razliku od modela iz prethodnog članka, modelirana kao razlika dviju simetričnih sigmoidnih funkcija. I u ovom se radu, pri usporedbi istih optimizacijskih metoda, algoritam strategije evolucije adaptacijom matrice kovarijance pokazao kao najbolji po metrici pokrivenosti područja.

U posljednjem radu istih autora vezanim za temu razmještaja osjetila [51], autori su ispitali prethodno opisane modele osjetila na visinskim kartama s preprekama koristeći algoritam simuliranog kaljenja, algoritam strategije evolucije adaptacijom matrice kovarijance te gradijentni spust (eng. Gradient Descent). U ovom je slučaju algoritam strategije evolucije adaptacijom matrice kovarijance dao bolje rezultate na manjim visinskim kartama, dok se kod većih visinskih karti gradijentni spust pokazao boljim. Dodatna prednost gradijentnog spusta je u postignutom značajnom ubrzanju vremenu izvođenja.

Kroz četiri rada Altahir i suradnici opisali su postupke modeliranja i razmještaja osje-

tila u dvodimenzionalnom prostoru na predefinirane pozicije. U radovima [62, 63] opisali su model usmjerenih osjetila binarne osjetilne sposobnosti i optimizaciju razmještaja za sigurnosne primjene, primarno za razmještaj sigurnosnih kamera. U radovima [64, 39] definicije osjetila proširili su općenitim opisima i dodali modele svesmjernih osjetila.

Thiene i suradnici su u radu [65] demonstrirali korištenje svesmjernih osjetila vjerojatnosne osjetilne sposobnosti za detekciju oštećenja u kompozitnim strukturama (eng. composite structures). Optimizacija razmještaja osjetila na dvodimenzionalnoj kompozitnoj strukturi provedena je koristeći genetski algoritam.

Binh je sa suradnicima u radu [66] usporedio niz optimizacijskih metoda za razmještaj svesmjernih osjetila u dvodimenzionalnom prostoru. Poboljšanu verziju kukavičjeg pretraživanja (eng. Improved Cuckoo Search Algorithm), algoritam kaotičnog oprašivanja cvijeća (eng. Chaotic Flower Pollination Algorithm), poboljšanu verziju genetskog algoritma (eng. Improved Genetic Algorithm) te algoritam optimizacije rojem čestica su usporedili na nizu testova te su pokazali da su prva dva algoritma, poboljšana verzija kukavičjeg pretraživanja i algoritam kaotičnog oprašivanja cvijeća, ostvarili bolje rezultate od ostalih i po pokrivenosti i po vremenu izvršavanja.

Liu Z. i suradnici su u radu [67] predložili jednostavne modele usmjerenih osjetila vjerojatnosne osjetilne sposobnosti te optimizirali razmještaj navedenih osjetila u dvodimenzionalnom prostoru na osnovu metrike pokrivenosti područja.

U radu [52] Krishnan i suradnici predložili su nekoliko pristupa razmještaja svesmjernih osjetila binarne osjetilne sposobnosti korištenjem k-pokrivenosti.

Hržić i suradnici, uključujući autora ove disertacije, u radu [68] opisali su rješenje razmještaja svesmjernih osjetila vjerojatnosnih osjetilnih sposobnosti u trodimenzionalnom prostoru. Rješenje za razmještaj motivirano je primjenom u lokalizaciji dronova u zatvorenom prostoru. Pri razmještaju osjetila korištena je metoda gradijentnog spusta za optimizaciju funkcije cilja temeljene na metrici pokrivenosti područja.

Kritter je sa suradnicima u radu [19] te u svojoj doktorskoj disertaciji [20] napravio pregled problema umjetničke galerije te povezao isti s problemom razmještaja osjetila, sigurnosnih kamera, u dvodimenzionalnom prostoru. Osjetila su bila usmjerena te modelirana koristeći binarnu osjetilnu sposobnost. Dodatno, predloženo rješenje omogućava korisniku označavanje bitnijih regija prostora te zadavanja njihovih težina.

Autor ove disertacije je sa suradnicima u radu [69] opisao problematiku razmještaja

osjetila za učinkovito pokrivanje dvodimenzionalnih i trodimenzionalnih prostora. U radu su predloženi pristupi modeliranju te modeli prostora i osjetilnih sposobnosti osjetila. Svesmjerna i usmjerena osjetila vjerojatnosnih osjetilnih sposobnosti razmještala su se koristeći tri genetska optimizacijska algoritma. Rezultati pokrivenosti pokazali su da je algoritam umjetnog roja pčela (eng. Artificial Bee Colony Algorithm) postigao bolje rezultate po metrici pokrivenosti prostora od algoritma optimizacije roja čestica i od algoritma vatrometa (eng. Fireworks Algorithm).

U tablici 2.1 dan je zbirni pregled prethodno opisanih radova uz odgovarajuće oznake korištenih vrsta prostora i osjetila te njihovih osjetilnih sposobnosti.

Tablica 2.1: Pregled vrsta i osjetilnih sposobnosti osjetila korištenih u analiziranim znanstvenim radovima

	Prostor			Osjetilna sposobnost		Vrsta osjetila	
	Dvodimenzionalni	Visinske karte	Trodimenzionalni	Binarna	Vjerojatnosna	Svesmjerno	Usmjereno
Dhillon i Chakrabarty [53]	✓			✓		✓	
Hörster i Lienhart [54]	✓			✓			✓
Hörster i Lienhart [55]	✓			✓			✓
Bodor i suradnici [18]	✓			✓			✓
Pålsson i Ståhl [56]	✓			✓			✓
Gonzalez-Barbosa i suradnici [57]	✓			✓		✓	✓
Hefeeda i Ahmadi [40]	✓				✓	✓	
Salehizadeh i suradnici [58]	✓			✓		✓	
Morsley i suradnici [59]	✓			✓		✓	✓
Akbarzadeh i suradnici [38]		✓		✓		✓	✓
Akbarzadeh i suradnici [60]		✓			✓		✓
Akbarzadeh i suradnici [61]		✓			✓		✓
Akbarzadeh i suradnici [51]		✓			✓		✓
Altahir i suradnici [62]	✓			✓		✓	
Altahir i suradnici [63]	✓			✓		✓	
Altahir i suradnici [64]	✓			✓		✓	✓
Altahir i suradnici [39]	✓			✓		✓	✓
Thiene i suradnici [65]	✓				✓	✓	
Liu Z. i suradnici [67]	✓				✓		✓
Binh i suradnici [66]	✓			✓		✓	
Krishnan i suradnici [52]	✓			✓		✓	
Hržić i suradnici [68]			✓		✓	✓	
Kritter i suradnici [19]	✓			✓			✓
Kritter [20]	✓			✓			✓
Sušanj i suradnici [69]	✓		✓		✓	✓	✓

3. Poglavlje

MODELIRANJE PROSTORA

Prvi znanstveni doprinos ovog istraživanja predloženi je model prostora. Kako bi se omogućilo određivanje razmještaja osjetila, potrebno je modelirati prostor u kojem se ista razmještaju. Model prostora koji se opisuje mora sadržavati topografske značajke prostora, poput njegovog oblika i prepreke koje se u njemu nalaze. Takav model mora čim vjerodostojnije opisati promatrani prostor, vodeći pritom računa o računskim mogućnostima i zahtjevima pri razmještaju.

Kao što je navedeno u poglavlju 1.3., kako bi se omogućio izračun pokrivenosti prostora osjetilima pri određivanju razmještaja osjetila, model prostora potrebno je podijeliti na konačan broj elemenata. Opisom prostora koji uzima u obzir veći broj detalja u topografiji prostora te podjelom istog na veći broj elemenata dobiveni model vjernije opisuje prostor, ali zbog povećanog broja elemenata i složenijeg opisa prostora izračun pokrivenosti predstavlja računski složeniju operaciju. Potrebno je omogućiti variranje broja elemenata koristeći diskretizacijsku značajku kako bi se postigao vjerni opis prostora uz prihvatljivu računsku složenost.

U dosadašnjim se istraživanjima prostor uglavnom modelirao jednostavnim poligonima u ravnini kao u primjerima problema umjetničke galerije. Dodatno, modelirao se i dvodimenzionalnim matricama kod kojih su numeričke vrijednosti opisivale radi li se o slobodnom (eng. free space), zauzetom (eng. occupied space) ili nepoznatom prostoru (eng. unknown space). Konačno, koristile su se i visinske karte, dvodimenzionalne matrice s vrijednostima koje opisuju visine prostora ili objekata u prostoru.

U ovom istraživanju predložen je model prostora te postupak modeliranja prostora. Ti

modeli prostora približno opisuju promatrani prostor kroz elemente slobodnog prostora i prepreka. Svaki element slobodnog prostora i prepreka opisan je baznim poligonom koji opisuje tlocrt elementa. U trodimenzionalnom modelu prostora taj se geometrijski lik omeđen baznim poligonom istisne u geometrijsko tijelo koje počinje i završava na zadanim visinama globalnog koordinatnog sustava modela prostora. Takvi elementi opisani baznim poligonima i visinama nazivaju se mreže (eng. mesh).

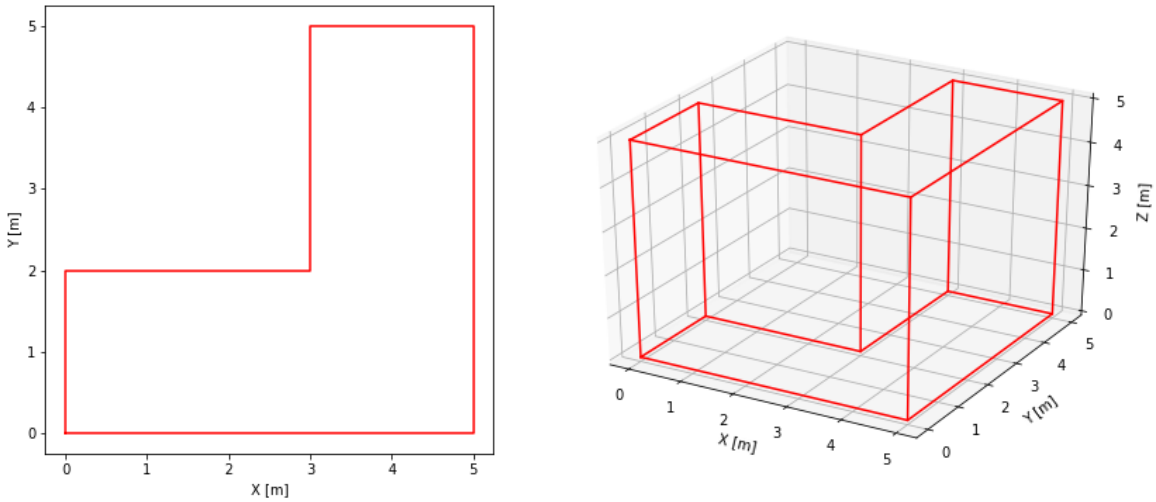
Skup svih elemenata slobodnog prostora i prepreka opisanih u zajedničkom globalnom koordinatnom sustavu predstavlja model prostora. Takav model prostora opisan je u trodimenzionalnom sustavu, ali ako postoji potreba za dvodimenzionalnim modelom prostora, poligoni koji opisuju prostor predstavljaju ujedno i njegov tlocrt te se mogu iskoristiti u tu svrhu. Predloženi pristup modeliranju omogućuje korištenje brzih izračuna osnovnih funkcija u izračunu vidljivosti. Te funkcije temelje se na računalnoj geometriji.

Prostori se sastoje od mreža raspoređenih u koordinatnom sustavu prostora čije je ishodište u pravilu u lijevom donjem kutu slobodnog prostora. Svaka mreža slobodnog prostora ili prepreke zadaje se s x i y koordinatama vrhova baznog poligona te početnom i završnom visinom. Poligon mora imati barem tri, ali može imati i veći broj vrhova. Koordinate vrhova poligona i visine mreže mogu poprimiti bilo koju vrijednost iz skupa realnih brojeva. Iz zadanih mreža slobodnog prostora i prepreka, postupkom rasterizacije određuju se točke koje se koriste za izračun pokrivenosti. Pri provjeri optičke vidljivosti između osjetila i točke u prostoru, koriste se sve mreže kojima je prostor opisan.

Na slici 3.1 dan je primjer jednostavnog prostora u obliku slova „L". Crvenim linijama označen je slobodan prostor. Na slici 3.1 a) prikazan je bazni poligon mreže slobodnog prostora, dok je na slici b) prikazana mreža navedenog slobodnog prostora.

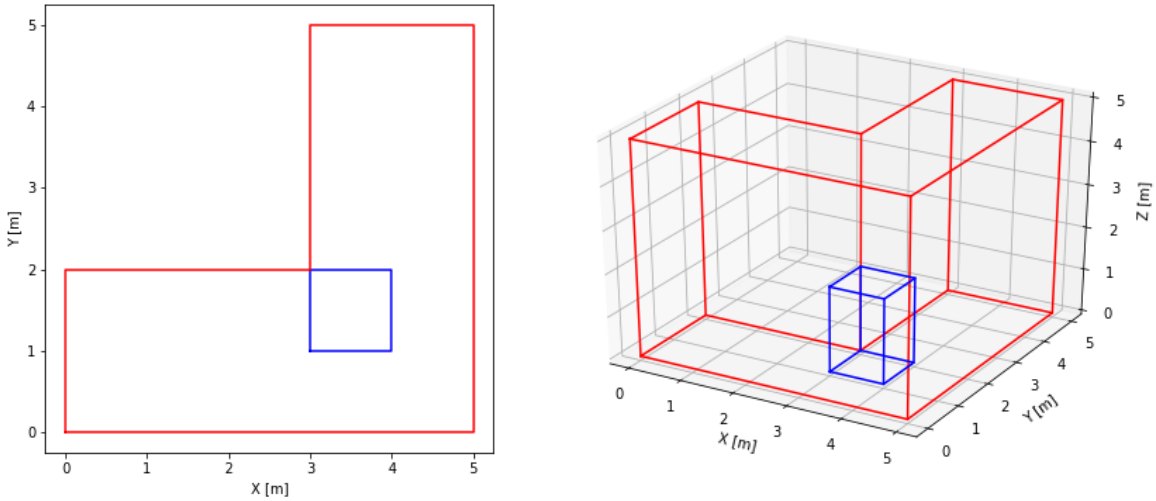
Na slici 3.2 prikazan je primjer iz slike 3.1 s dodanom preprekom označenom plavom bojom. Prepreka je također opisana mrežom s pripadajućim baznim poligonom. Na slici se ta prepreka nalazi unutar slobodnog prostora, ali prekriva samo njegov manji dio.

Ovakav postupak modeliranja prostora pruža dovoljnu slobodu pri izradi modela prostora te omogućava izradu modela prostora koji približno dobro opisuju promatrani prostora. Primjer jednog složenijeg prostora dan je na slici 3.3.



(a) (b)

Slika 3.1: Primjer jednostavnog prostora bez prepreka

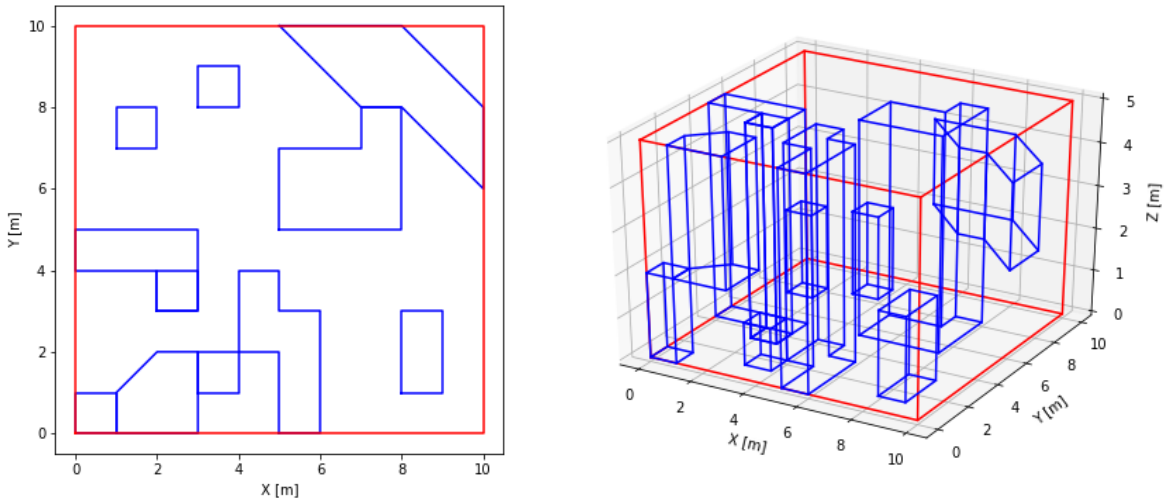


(a) (b)

Slika 3.2: Primjer jednostavnog prostora s preprekom

3.1. Provjera ispravnosti pozicije i optičke vidljivosti

Predloženi pristup modeliranju prostora omogućuje učinkovite provjere ispravnosti bilo koje pozicije u prostoru te provjeru optičke vidljivosti. Prilikom rasterizacije prostora



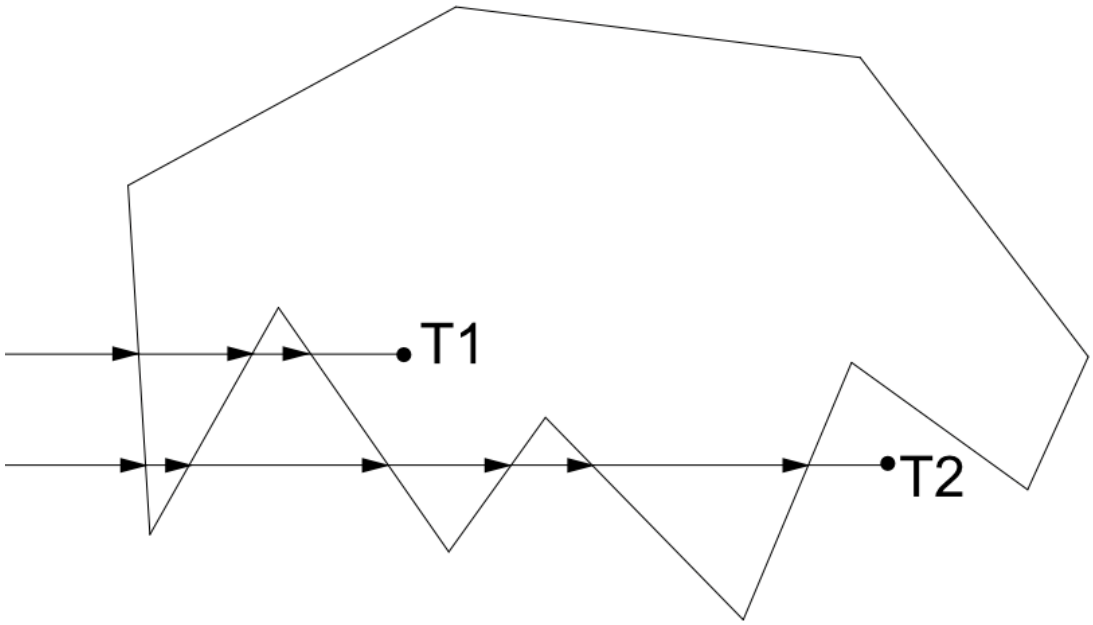
(a) (b)

Slika 3.3: Primjer složenog prostora

i provjeri optičke vidljivosti potrebno je provjeriti je li točka unutar područja slobodnog prostora ili izvan njega te je li dio prepreke. Za točke unutar područja slobodnog prostora kažemo da imaju ispravnu poziciju, odnosno da je točka ispravna.

Svaka točka u prostoru opisana je s vrijednostima koordinata iz skupa realnih brojeva. Provjera ispravnosti pozicije se za odabranu točku prostora izvršava na svim mrežama koje opisuju model prostora. Za svaku od mreža slobodnog prostora i prepreka za koju se radi provjera, provjerava se ako je točka unutar baznog poligona mreže koristeći algoritam praćenja zrake (eng. Ray-tracing Algorithm) [70]. U osnovi, algoritam praćenja zrake broji sjecišta zrake, uglavnom horizontalne linije, koja prolazi kroz zadanu točku te poligona. Na slici 3.4 dan je primjer algoritma praćenja zrake, gdje se za horizontalne linije povučene od lijevog ruba, izvan poligona, do točaka $T1$ i $T2$ broje sjecišta tih horizontalnih linija s poligonom. Ako je broj sjecišta neparan točka se nalazi unutar poligona ($T1$), dok se za slučaj parnog broja sjecišta točka nalazi izvan poligona ($T2$). Dodatno, ako se radi o trodimenzionalnom modelu prostora, potrebno je provjeriti i je li z koordinata točke unutar zadanih visina mreže.

Provjera je li točka ispravna prvo se provodi nad mrežama slobodnog prostora. Ako točka nije dio niti jedne mreže slobodnog prostora, njena se pozicija smatra neispravnom. Kada se pronađe prva mreža slobodnog prostora koja sadrži točku prelazi se na provjeru



Slika 3.4: Algoritam praćenja zrake za provjeru ispravnosti pozicije

je li točka dio neke od mreža prepreka. Ako se utvrdi da je točka dio neke od mreža prepreka, tada se njena pozicija također smatra neispravnom. Kada je provjera završila za sve mreže prepreka i utvrdilo se da je točka dio barem jedne mreže slobodnog prostora i da nije dio niti jedne od mreža prepreka, njena je pozicija ispravna.

Točke koje se nalaze na rubovima slobodnog prostora smatraju se dijelom prostora, dok one koje se nalaze na rubovima prepreka nisu dio istog. Pseudokod algoritma za provjeru ispravnosti pozicije dan je u algoritmu 3.1.

Algoritam 3.1: Provjera ispravnosti točke prostora

```
1  FUNCTION point_inside_polygon(point, polygon)
2      IF polygon CONTAINS point
3      THEN
4          RETURN True
5      END IF
6
7      RETURN False
8  END
9
10 FUNCTION point_inside_mesh(point, mesh)
11     SET poly_flag TO point_inside_polygon(point, mesh.polygon)
12
13     IF mesh.dimension IS 3
14     THEN
15         SET poly_flag TO poly_flag AND mesh.bottom < point.z < mesh.top
```

```
16      END IF
17
18      RETURN poly_flag
19  END
20
21  FUNCTION point_valid(point, environment)
22      SET valid_flag TO False
23
24      FOR mesh IN environment.positive_meshes
25      DO
26          IF point_inside_mesh(point, mesh)
27          THEN
28              SET valid_flag TO True
29              BREAK
30          END IF
31      END FOR
32
33      IF NOT valid_flag
34      THEN
35          RETURN False
36      END IF
37
38      FOR mesh IN environment.negative_meshes
39      DO
40          IF point_inside_mesh(point, mesh)
41          THEN
42              RETURN False
43          END IF
44      END FOR
45
46      RETURN True
47  END
```

Za par točaka p_1 i p_2 između kojih je potrebno utvrditi postojanje optičke vidljivosti prvo se provjerava ispravnost njihovih pozicija koristeći ranije opisani postupak. Ako su obje točke ispravne, odnosno dio slobodnog prostora, odrede se projekcije tih točaka na tlocrt modela prostora. Kako se radi o projekcijama na tlocrt modela prostora, koordinate projekcija točaka odgovaraju x i y koordinatama samih točaka dok $z = 0$. Temeljem tih točaka odredi se jednadžba pravca p_s koji prolazi kroz njih:

$$y - p_{1_y} = \frac{p_{2_y} - p_{1_y}}{p_{2_x} - p_{1_x}} \cdot (x - p_{1_x}). \tag{3.1}$$

Potom se provjerava presijeca li navedeni pravac neki od baznih poligona mreža kojima

je opisan prostor, a u domeni između dviju točaka. Ako pravac presijeca neki od baznih poligona mreža, bilo slobodnog prostora ili prepreka, za dvodimenzionalne modele prostora optička vidljivost između navedene dvije točke ne postoji. Kod trodimenzionalnih se modela prostora odrede koordinate sjecišta pravca p_s i baznih poligona mreža te se odredi vrijednost p_{s_z} koordinate točke sjecišta:

$$p_{s_z} = p_{1_z} + (p_{2_z} - p_{1_z}) \cdot \frac{p_{s_x} - p_{1_x}}{p_{2_x} - p_{1_x}}. \tag{3.2}$$

Ako je pravac p_s paralelan s ravninom koja sadrži osi ordinate y i aplikate z, u desnom dijelu gornjeg izraza se umjesto x koordinata analogno koriste y koordinate točaka.

Ako se izračunata vrijednost p_{s_z} nalazi unutar visina kojima je opisana mreža, tada pravac presijeca mrežu i u trodimenzionalnom modelu prostora. Ako ne postoje sjecišta pravca i baznih poligona ili je visina pravca u sjecištu izvan mreže, tada optička vidljivost između dvije točke postoji.

Predloženim pristupom koji koristi brze implementacije funkcija računalne geometrije u dvodimenzionalnoj projekciji prostora omogućava se učinkovita provjera postojanja optičke vidljivosti. U algoritmu 3.2 dan je pseudokod za provjeru optičke vidljivosti između dvije točke.

Algoritam 3.2: Provjera optičke vidljivosti

```
1  FUNCTION line_intersects_polygon(point1, point2, polygon)
2      SET line TO line passing through the points
3
4      IF line INTERSECTS polygon
5          RETURN list of intersections
6      END IF
7
8      RETURN empty list
9  END
10
11 FUNCTION line_intersects_mesh(point1, point2, mesh)
12     IF point_inside_polygon(point1, mesh.polygon) XOR point_inside_polygon(point2,
           mesh.polygon)
13     THEN
14         RETURN True
15     END IF
16
17     SET intersections TO line_intersects_polygon(point1, point2, mesh.polygon)
18
```

```
19      IF intersections IS NOT empty list
20      THEN
21          IF mesh.dimension IS 3
22          THEN
23              FOR i IN intersections
24              DO
25                  IF point1.x IS NOT point2.x
26                  THEN
27                      SET z TO point1.z + (point2.z - point1.z) * ((i.x - point1.x) / (
                            point2.x - point1.x))
28                  ELSE IF point1.y IS NOT point2.y
29                  THEN
30                      SET z TO point1.z + (point2.z - point1.z) * ((i.y - point1.y) / (
                            point2.y - point1.y))
31                  END IF
32
33                  IF mesh.bottom <= z <= mesh.top
34                  THEN
35                      RETURN True
36                  END IF
37              END FOR
38      ELSE
39          RETURN True
40      END IF
41
42      IF mesh.dimension IS 3 AND point_inside_polygon(point1, mesh.polygon) AND
            point_inside_polygon(point2, mesh.polygon)
43      THEN
44          IF point1.z > mesh.top AND point1.z < mesh.bottom OR point2.z > mesh.top AND
                point1.z < mesh.bottom
45          THEN
46              RETURN True
47          END IF
48      END IF
49
50      RETURN False
51  END
52
53  FUNCTION line_of_sight(point1, point2, environment)
54      FOR mesh IN environment.positive_meshes
55      DO
56          IF line_intersects_mesh(point1, point2, mesh)
57          THEN
58              RETURN False
59          END IF
60      END FOR
61
62      FOR mesh IN environment.negative_meshes
```

```
63      DO
64          IF line_intersects_mesh(point1, point2, mesh)
65          THEN
66              RETURN False
67          END IF
68      END FOR
69
70      RETURN True
71  END
```

3.2. Rasterizacija prostora

Kako je navedeno ranije, slobodni prostor se diskretizira na konačan broj elemenata, a kako bi se omogućio izračun pokrivenosti prostora koristeći odabranu metriku pokrivenosti. Ta se podjela vrši koristeći rasterizacijsku vrijednost definiran kao dužina jedne stranice voksela. Kako je navedeno ranije, korisniku se omogućava promjena rasterizacijske vrijednosti, a s ciljem balansiranja vjernijeg opisa prostora većim brojem voksela i dostupnih računalnih resursa.

Broj voksela u prostoru ne ovisi samo o dimenzionalnosti, veličini i rasporedu slobodnog prostora i prepreka, već i o zadanoj rasterizacijskoj vrijednosti. Smanjenjem rasterizacijske vrijednosti povećava se i broj voksela u prostoru te se time omogućava vjerniji opis prostora.

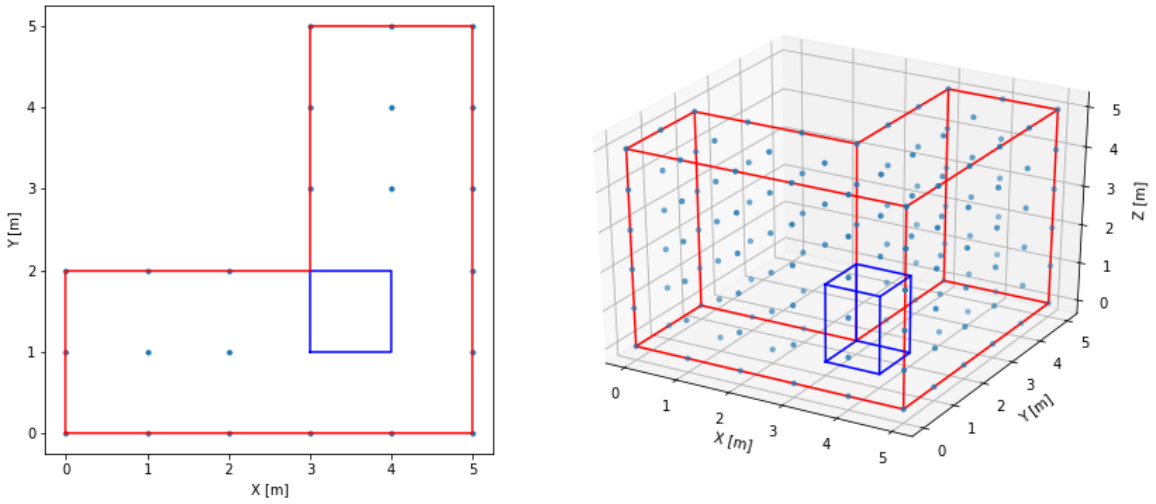
Prvi korak pri rasterizaciji prostora je određivanje granica prostora. Granice prostora uzimaju najmanje x_{min}, y_{min} i z_{min} te najveće vrijednosti granica mreža x_{max}, y_{max} i z_{max} slobodnog prostora po svakoj od koordinata. Granice svake od mreža sastoje se od najmanje i najveće vrijednosti x i y koordinata točaka kojima je poligon opisan te od donje i gornje visine z mreže. Unutar šest određenih granica prostora uzimaju se točke s korakom koji odgovara rasterizacijskoj vrijednosti za svaku od koordinata prostora. Za svaku točku opisanu skupom tako dobivenih koordinata provjerava se ispravnost njene pozicije te, ako je pozicija ispravna, voksel opisan tom pozicijom dodaje se u popis voksela slobodnog prostora. Odabrane pozicije predstavljaju centralnu točku odgovarajućeg voksela. Pseudokod rasterizacije prostora dan je u algoritmu 3.3.

Algoritam 3.3: Rasterizacija prostora

```
1  FUNCTION mesh_bounds(mesh)
2      SET x_coords TO empty list
3      SET y_coords TO empty list
4
5      FOR point in mesh.polygon.points
6      DO
7          APPEND point.x TO x_coords
8          APPEND point.y TO y_coords
9      END FOR
10
11     RETURN min(x_coords), max(x_coords), min(y_coords), max(y_coords), mesh.bottom,
           mesh.top
12 END
13
14 FUNCTION environment_bounds(environment)
15     SET x_coords TO empty list
16     SET y_coords TO empty list
17     SET z_coords TO empty list
18
19     FOR mesh IN environment.positive_meshes
20     DO
21         SET x_min, x_max, y_min, y_max, z_min, z_max TO mesh_bounds(mesh)
22
23         APPEND x_min TO x_coords
24         APPEND x_max TO x_coords
25         APPEND y_min TO y_coords
26         APPEND y_max TO y_coords
27         APPEND z_min TO z_coords
28         APPEND z_max TO z_coords
29     END FOR
30
31     RETURN min(x_coords), max(x_coords), min(y_coords), max(y_coords), min(z_coords),
           max(z_coords)
32 END
33
34 FUNCTION generate_voxels(rasterization, environment)
35     SET voxels TO empty list
36
37     SET x_min, x_max, y_min, y_max, z_min, z_max TO environment_bounds(environment)
38
39     SET x to x_min
40     WHILE x <= x_max:
41     DO
42         SET y to y_min
43         WHILE y <= y_max
44         DO
45             SET z to z_min
```

```
46              WHILE z <= z_max
47              DO
48                  SET point TO x, y, z
49
50                  IF point_valid(point, environment)
51                      APPEND point TO voxels
52                  END IF
53
54                  SET z TO z + rasterization
55              END WHILE
56              SET y TO y + rasterization
57          END WHILE
58          SET x TO x + rasterization
59      END WHILE
60
61      RETURN voxels
62  END
```

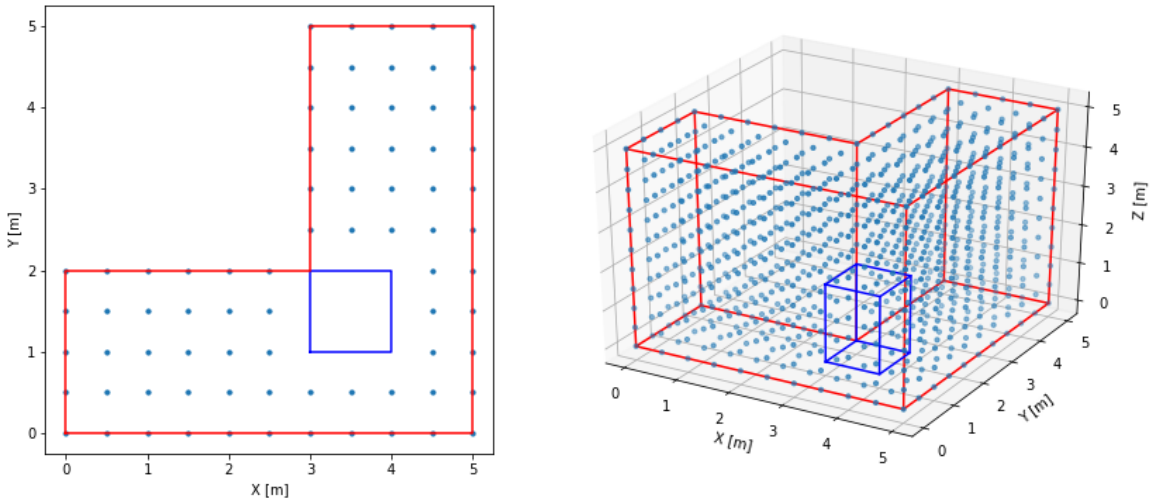
Na sljedećim slikama prikazane su generirane točke ranije opisanog jednostavnog prostora s preprekom postupkom rasterizacije za različite rasterizacijske vrijednosti. Na slici 3.5 prikazane su točke generirane rasterizacijskom vrijednosti $r_v = 1{,}0\,m$ za dvodimenzionalni i trodimenzionalni model prostora. Na slici 3.6 dan je prikaz dobiven rasterizacijskom vrijednosti $r_v = 0{,}5\,m$, a na slici 3.7 prikaz za rasterizacijsku vrijednost $r_v = 0{,}2\,m$.



(a) (b)

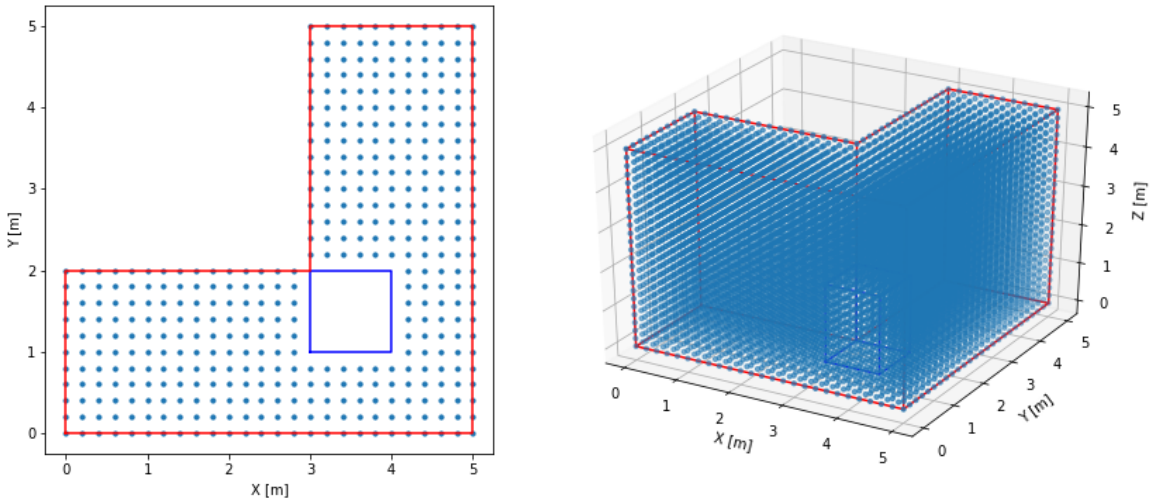
Slika 3.5: Primjer generiranih točaka za vrijednost rasterizacije $r_v = 1{,}0\,m$

Koristeći dvodimenzionalne i trodimenzionalne modele prostora kao i različite raste-



(a) (b)

Slika 3.6: Primjer generiranih točaka za vrijednost rasterizacije $r_v = 0{,}5\,m$



(a) (b)

Slika 3.7: Primjer generiranih točaka za vrijednost rasterizacije $r_v = 0{,}2\,m$

rizacijske vrijednosti, omogućava se vrednovanje performansi optimizacijskih algoritama na ispitnim slučajevima različitih složenosti.

4. Poglavlje

MODELIRANJE OSJETILNIH SPOSOBNOSTI OSJETILA

Drugi znanstveni doprinos ovog istraživanja prijedlog je generičkih modela osjetilnih sposobnosti osjetila te prijedlog modela osjetilnih sposobnosti svesmjernog i usmjerenog osjetila. Kao što je vidljivo u poglavlju 2., dosadašnja istraživanja koristila su ili binarne ili jednostavne vjerojatnosne modele osjetilnih sposobnosti.

U kontekstu ovog istraživanja, osjetilima se smatraju svi uređaji koji odašilju ili primaju signal. Na temelju osnovnih modela osjetilnih sposobnosti svesmjernog i usmjerenog osjetila izrađeni su modeli osjetilnih sposobnosti predstavnika tih osjetila. Kao predstavnik svesmjernog osjetila predložen je model odašiljača radio-frekvencijskog signala, a kao predstavnik usmjerenog osjetila dubinska stereo kamera.

Predloženi modeli osjetilnih sposobnosti osjetila opisani su nizom pravila, odnosno matematičkih funkcija, za izračun vidljivosti za svaki od voksela prostora u okolini osjetila. Predloženi modeli osjetila temelje se na vjerojatnosnom modelu pokrivanja koji za svaki voksel prostora u okolini osjetila određuje vrijednost vidljivosti u rasponu od nula do jedan. Osjetilna sposobnost i opis osnovnih funkcija vidljivosti opisane su u trodimenzionalnom sustavu, a u nastavku su navedene i funkcije koje se razlikuju za dvodimenzionalni sustav.

Svaki voksel v_o prostora u okolini osjetila opisan je svojim koordinatama v_{o_x}, v_{o_y} i v_{o_z} u Kartezijevom koordinatnom sustavu s ishodištem u točki osjetila. Vidljivost $v_v(v_o)$

voksela v_o definira se kao umnožak triju osnovnih funkcija:

$$v_v(v_o) = v_d(v_o) \cdot v_\varphi(v_o) \cdot v_\theta(v_o), \tag{4.1}$$

gdje je $v_d(v_o)$ funkcija vidljivosti koja ovisi o udaljenosti d između osjetila i voksela, dok su funkcije $v_\varphi(v_o)$ i $v_\theta(v_o)$ ovisne o kutevima između osjetila i voksela. Funkcija $v_\varphi(v_o)$ opisuje ovisnost vidljivosti o kutu azimuta (eng. azimuth) φ prema vokselu, dok vrijednost funkcije $v_\theta(v_o)$ ovisi o kutu inklinacije (eng. inclination) θ prema odredišnom vokselu.

Vrijednosti svih triju funkcija, $v_d(v_o)$, $v_\varphi(v_o)$ te $v_\theta(v_o)$ su u rasponu između nula i jedan. Kada je vrijednost jedne od funkcija nula, odredišni voksel nije vidljiv, a ako je vrijednost svih triju funkcija jedan, voksel je potpuno vidljiv. Kako je rezultantna funkcija vidljivosti $v_v(v_o)$ umnožak triju funkcija, njena vrijednost se nalazi u istom rasponu. Za sve četiri funkcije vrijedi:

$$v_v(v_o),\, v_d(v_o),\, v_\varphi(v_o),\, v_\theta(v_o) \in [0, 1]\,. \tag{4.2}$$

Definicija svake od triju funkcija vidljivosti ovisi o osjetilu koje se opisuje. Funkcije vidljivosti koriste osnovne funkcije za izračune udaljenosti, kuta azimuta te kuta inklinacije između odredišnog voksela i osjetila. Udaljenost $d_v(v_o)$ između voksela i osjetila koristi se pri izračunu funkcije vidljivosti koja ovisi o udaljenosti $v_d(v_o)$ te se izračunava koristeći definiciju Euklidske udaljenosti:

$$d_v(v_o) = \sqrt{v_{o_x}^2 + v_{o_y}^2 + v_{o_z}^2}, \tag{4.3}$$

za trodimenzionalni model, odnosno:

$$d_v(v_o) = \sqrt{v_{o_x}^2 + v_{o_y}^2}, \tag{4.4}$$

za dvodimenzionalni model prostora.

Oba kuta, azimut φ i inklinacija θ, izračunavaju se kao kutevi sfernog koordinatnog sustava. Vrijednost kuta azimuta izračunava se vodeći računa o kvadrantu u kojem se

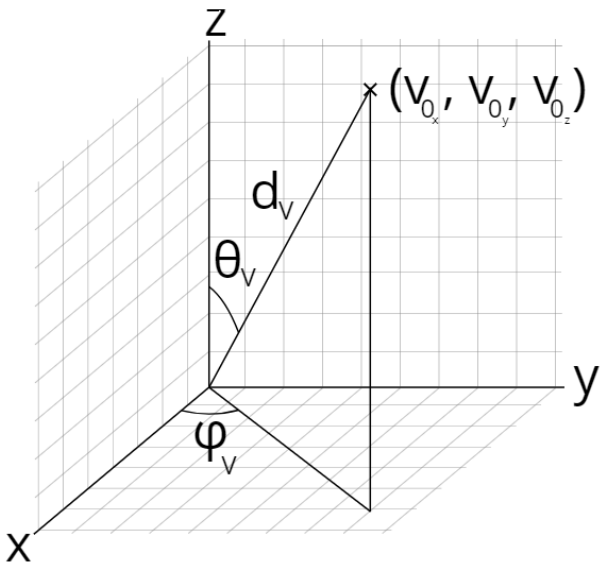
nalaze vrijednosti x i y koordinata:

$$\varphi_v(v_o) = \arctan 2\left(v_{o_y}, v_{o_x}\right). \tag{4.5}$$

Pri izračunu kuta inklinacije koristi se prethodno izračunata vrijednost udaljenosti između voksela i osjetila $d_v(v_o)$:

$$\theta_v(v_o) = \arcsin\left(\frac{v_{o_z}}{d_v(v_o)}\right). \tag{4.6}$$

Na slici 4.1 prikazane su vrijednosti udaljenosti d_v te kuteva φ_v i θ_v ka vokselu v_0 u koordinatnom sustavu osjetila.



Slika 4.1: Primjer udaljenosti d_v, kuteva φ_v i θ_v za voksel u koordinatnom sustavu osjetila

4.1. Model odašiljača radio-frekvencijskog signala

Svesmjerna osjetila odašilju ili primaju signal u svim smjerovima, neovisno o smjeru signala, odnosno kutu između odredišnog voksela i referentnog osjetila. Jedni od najčešćih predstavnika svesmjernih osjetila su odašiljači radio-frekvencijskog signala. Ti odašiljači sadrže svesmjernu antenu te, u idealnom slučaju, odašilju signal jednake snage u svim smjerovima. U nastavku je predložen i opisan općenit i pojednostavljen model odašiljača radio-frekvencijskog signala. Takav model ne uključuje utjecaje refleksije od površina u

prostoru ili apsorpcije valova od strane objekata u prostoru.

Sva osjetila sa svesmjernom antenom imaju konstantnu kutnu vidljivost prema odredišnom vokselu. Ta kutna vidljivost je definirana kao:

$$v_{\varphi}(v_o) = 1, \tag{4.7}$$

za vidljivost ovisnu o kutu azimuta te kao:

$$v_{\theta}(v_o) = 1, \tag{4.8}$$

za vidljivost ovisnu o kutu inklinacije.

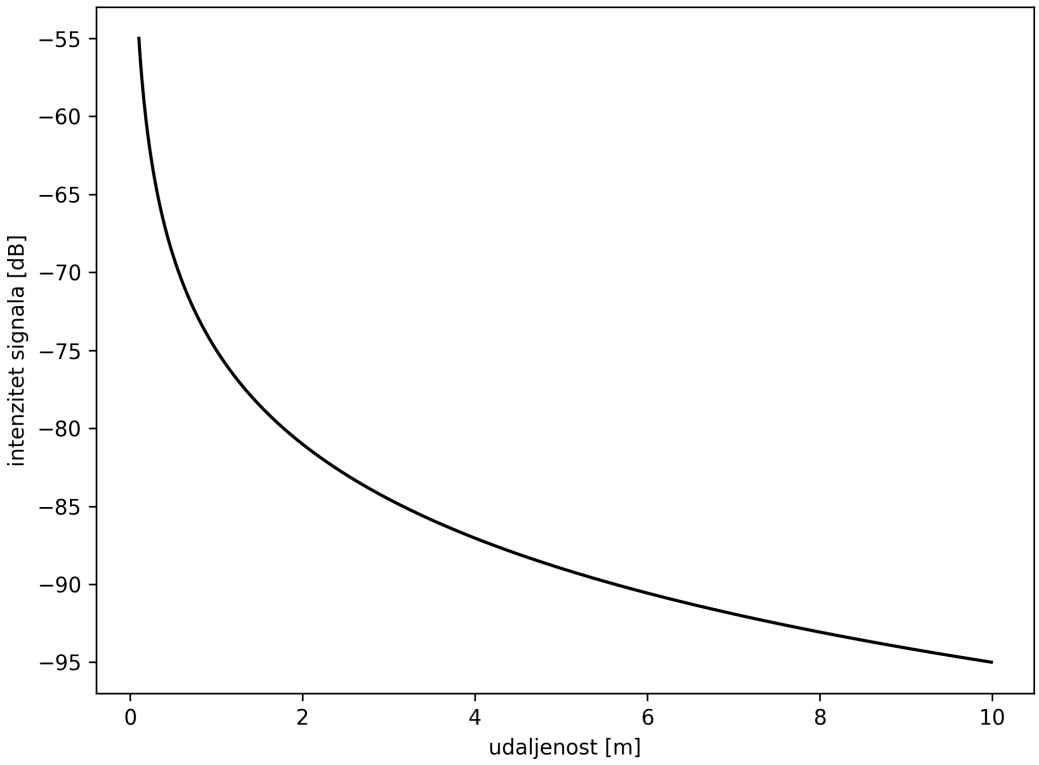
Vidljivost ovisna o udaljenosti $v_d(v_o)$ za ovu vrstu osjetila opisana je koristeći funkciju intenziteta primljenog signala (eng. Received Signal Strength Indicator - RSSI) [71]. Rezultantna vrijednost u decibelima predstavlja odnos intenziteta signala na mjernoj udaljenosti te intenziteta signala na poznatoj udaljenosti $rssi_{1m}$:

$$rssi(v_o) = -20 \cdot log_{10}(d_v(v_o)) + rssi_{1m}. \tag{4.9}$$

Funkcija intenziteta primljenog signala u ovisnosti o udaljenosti od osjetila prikazana je na slici 4.2.

Kako je vidljivo na slici 4.2, intenzitet signala opada s povećanjem udaljenosti od osjetila. U jednom trenutku, na zadanoj udaljenosti, intenzitet signala padne ispod intenziteta šuma. Razina intenziteta šuma predstavlja vrijednost ispod koje se signal ne može raspoznati u odnosu na ostatak radio-frekvencijskog zračenja u prostoru. Kako razina intenziteta šuma predstavlja najmanju iskoristivu vrijednost intenziteta signala, pri izračunu funkcije vidljivosti koristi se odnos signal-šum (eng. signal-to-noise ratio - SNR). Odnos signal-šum predstavlja razliku između intenziteta primljenog signala $rssi(v_o)$ i vrijednosti intenziteta šuma *noise* u decibelima. Kada je vrijednost intenziteta primljenog signala jednaka ili manja vrijednosti intenziteta šuma, rezultantna vrijednost funkcije odnosa signal-šum postavlja se na nula decibela:

$$snr(v_o) = \begin{cases} rssi(v_o) - noise, & \text{akko } rssi(v_o) > noise, \\ 0, & \text{inače.} \end{cases} \tag{4.10}$$



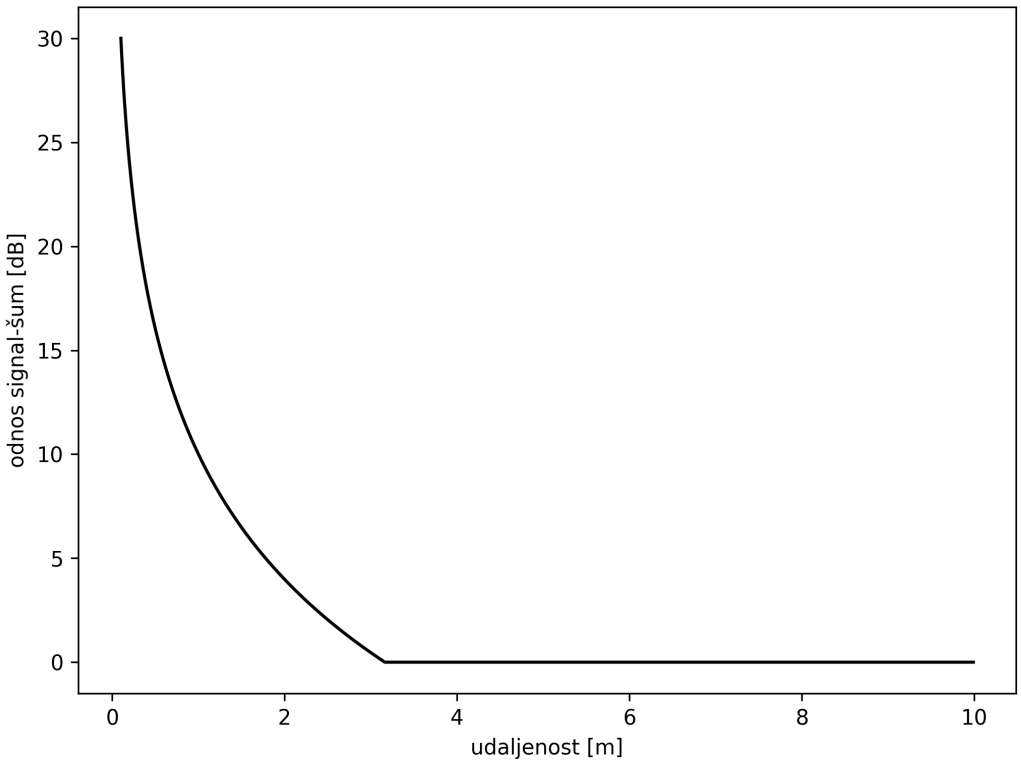
Slika 4.2: Primjer funkcije intenziteta primljenog signala $rssi(v_o)$ u ovisnosti o udaljenosti od osjetila

Funkcija odnosa signal-šum $snr(v_o)$ u ovisnosti o udaljenosti od osjetila prikazana je na slici 4.3.

Izračun vrijednosti vidljivosti, formula 4.1, temelji se na pretpostavci da će vrijednost funkcije vidljivosti u ovisnosti o udaljenosti biti u domeni $v_d(v_o) \in [0, 1]$. Da bi se navedena pretpostavka zadovoljila potrebno je normalizirati vrijednosti funkcije odnosa signal-šum koristeći vrijednost $rssi_{max}$, vrijednost intenziteta primljenog signala u točki najbližoj osjetilu:

$$v_d(v_o) = \frac{snr(v_o)}{rssi_{max} - noise}. \tag{4.11}$$

Ovako normaliziran odnos signal-šum, slika 4.4, predstavlja vrijednost funkcije vidljivosti u ovisnosti o udaljenosti $v_d(v_o)$ od osjetila.



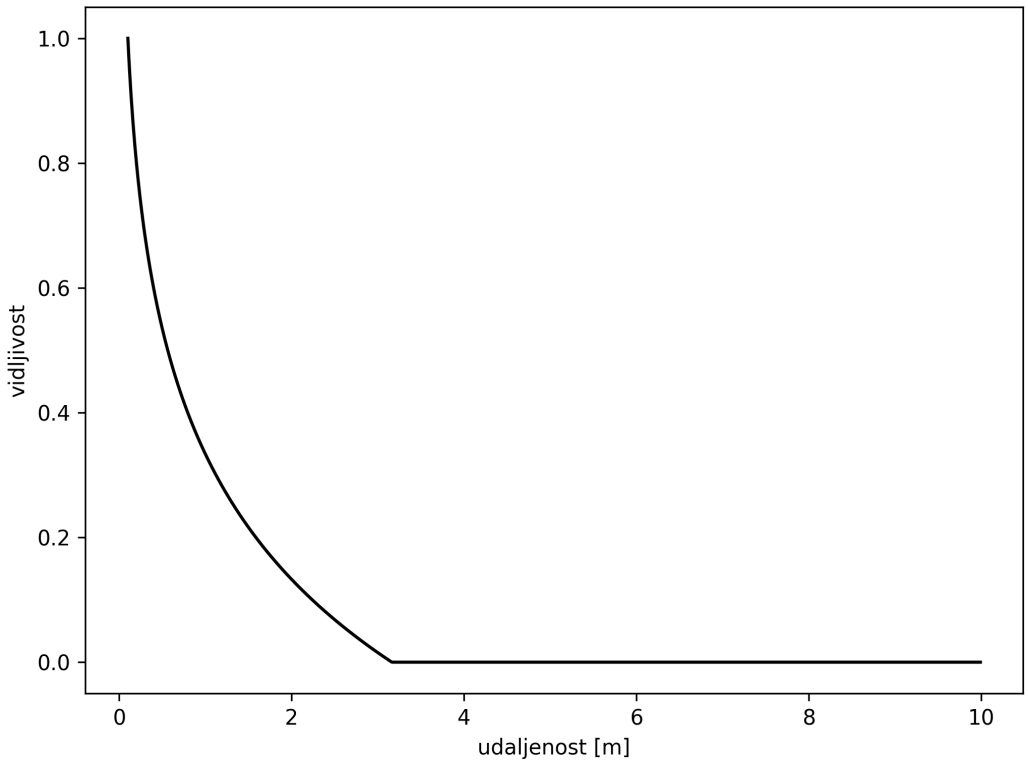
Slika 4.3: Primjer funkcije odnosa signal-šum $snr(v_o)$ u ovisnosti o udaljenosti od osjetila

4.2. Model dubinske stereo kamere

Druga kategorija predloženih modela osjetila odnosi se na usmjerena osjetila. Usmjerena osjetila su ona kod kojih vrijednost funkcije vidljivosti između osjetila i voksela u prostoru $v_v(v_o)$ ovisi ne samo o udaljenosti od osjetila, već i o kutevima azimuta i inklinacije. Uobičajeni predstavnici ove kategorije su razne video kamere.

Model predstavnika usmjerenih osjetila predložen u ovom istraživanju temelji se na stereo kameri i postupku stereo usporedbe (eng. Stereo Matching) [72]. Stereo kamera je sustav dviju kamera unutar istog kućišta. Takve kamere uobičajeno leže u istoj ravnini na zadanoj međusobnoj udaljenosti (eng. baseline distance). Osnovna primjena stereo kamera je određivanje trodimenzionalnih značajki iz dvije slike, primjerice udaljenosti točke prostora od stereo kamere. Ta se udaljenost, odnosno dubina, izračunava kroz postupak stereo usporedbe.

Prije izvođenja postupka stereo usporedbe, potrebno je odrediti međusobni odnos dviju

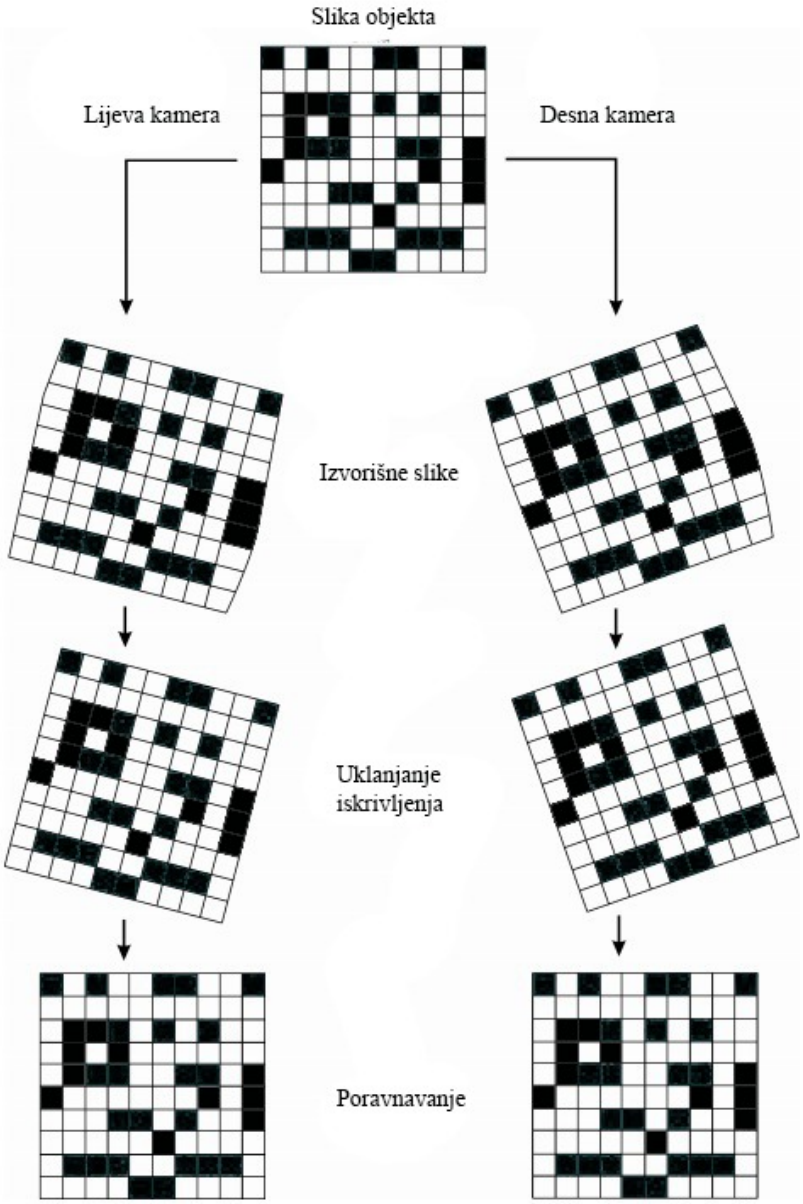


Slika 4.4: Primjer funkcije vidljivosti u ovisnosti o udaljenosti $v_d(v_o)$ za odašiljače radiofrekvencijskog signala

kamera koristeći postupak umjeravanja [73, 74]. Umjeravanje je postupak koji se uglavnom izvodi samo jednom po osjetilu i uobičajeno se provede koristeći potpuno ravnu ploču s iscrtanim uzorkom, uglavnom šahovske ploče poznatih dimenzija i broja polja ili većim brojem koncentričnih kružnica poznatih promjera. Iz poznatih relativnih odnosa točaka koje je moguće detektirati u uzorku, odredi se relativan odnos dviju kamera, odnosno odrede se ekstrinzični značajke stereo kamere [74]. Postupkom umjeravanja moguće je odrediti i distorziju slike, odnosno intrinzične značajke, svake od kamera.

Stereo usporedba je postupak koji se uobičajeno radi na poravnatim slikama iz obje kamere. Postupkom poravnavanja (eng. rectification) [75] slika svake od kamera se izobliči, rotira, skalira i slično, kako bi se parovi piksela koji odgovaraju istim poznatim točkama uvijek nalazili na istoj epipolarnoj liniji (eng. epipolar line), odnosno u istom retku u obje slike. Postupak poravnavanja prikazan je primjerom na slici 4.5.

Vrijednosti koje opisuju relativan odnos para poravnatih slika sačuvan je u formi Q-



Slika 4.5: Primjer postupka poravnavanja slike

matrice:

$$Q = \begin{bmatrix} 1 & 0 & 0 & -c_x \\ 0 & 1 & 0 & -c_y \\ 0 & 0 & 0 & f \\ 0 & 0 & \frac{-1}{T_x} & \frac{c_x - c_x^{'}}{T_x} \end{bmatrix}. \tag{4.12}$$

Vrijednost T_x je udaljenost između kamera, zapisan u metrima ili milimetrima. Vrijednosti c_x i c_y koordinate su centralne točke (eng. principal point) C referentne kamere. Za referentnu se kameru najčešće uzima lijeva kamera te se sve izračunate vrijednosti

dubine poravnavaju u odnosu na nju. Vrijednost $c_x^{'}$ je x koordinata centralne točke nereferentne, uglavnom desne, kamere. Sve vrijednosti koordinata centralnih točki zapisane su u pikselima. Kada su obje kamere iste rezolucije i ako je proveden standardan postupak poravnavanja, c_x i $c_x^{'}$ će imati istu vrijednost te će posljednji element Q-matrice biti nula. Vrijednost f sadrži žarišnu duljinu (eng. focal length) u pikselima.

Ključni korak u procesu stereo usporedbe je pronalazak sličnih piksela u obje slike. Ta dva piksela, u idealnom slučaju, odgovaraju reprezentaciji iste točke u prostoru scene [76]. Pretraga za sličnim pikselima provodi se samo do najveće udaljenosti d_{max} od referentnog piksela u jednom smjeru. Pretraga se ograničava na d_{max} zbog mogućnosti nepostojanja piksela koji odgovaraju istoj točki scene, a zbog niza razloga, primjerice zbog okluzije (eng. occlusion). Ako su pronađeni slični pikseli, apsolutna udaljenost između njihovih x_e i x_d koordinata se koristi za izračun udaljenosti te točke scene te se zove disparitet (eng. disparity):

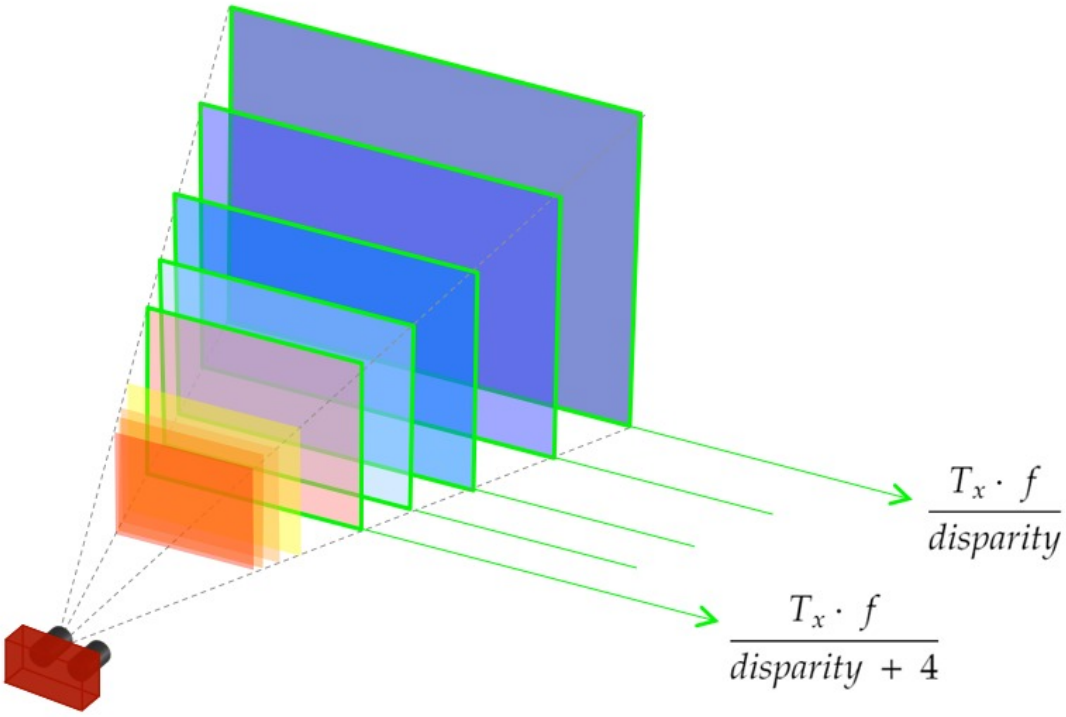
$$disparity = |x_e - x_d| \,. \tag{4.13}$$

Iz vrijednosti dispariteta i vrijednosti dobivenih iz Q-matrice, udaljenost točke scene *depth* od ravnine stereo kamere može se izračunati koristeći:

$$depth = \frac{T_x \cdot f}{disparity}. \tag{4.14}$$

Na slici 4.6 prikazane su razine dispariteta u ovisnosti o udaljenosti od kamere te primjer razlike u njihovoj udaljenosti. Za razine dispariteta bliže kameri dubina je manja, a vrijednost dispariteta, odnosno udaljenost dva slična piksela, je veća.

Kako je razina dispariteta diskretna vrijednost, pri daljnjem izračunu funkcije vidljivosti uvodimo funkciju pogreške definiranu kao razliku između dvije uzastopne razine dispariteta. Vokseli se u prostoru uglavnom nalaze između dvije uzastopne diskretne vrijednosti dubine, odnosno dviju razina dispariteta. Prilikom stereo usporedbe odabire se jedan od dva susjedna piksela te se time uvodi pogreška između stvarne dubine i dubine dobivene postupkom stereo usporedbe. U ovom se modelu pogreška definira kao udaljenost diskretnih razina dubine oko poznate udaljenosti voksela i opisuje se funkcijom pogreške $error_d(v_o)$. Za voksel na udaljenosti $d_v(v_o)$ od osjetila vrijednost funkcije pogreške se



Slika 4.6: Primjer razina dispariteta [72]

izračunava kao udaljenost dviju razina dispariteta između kojih se voksel nalazi:

$$error_d(v_o) = \frac{T_x \cdot f}{\lfloor disparity \rfloor} - \frac{T_x \cdot f}{\lfloor disparity + 1 \rfloor}. \tag{4.15}$$

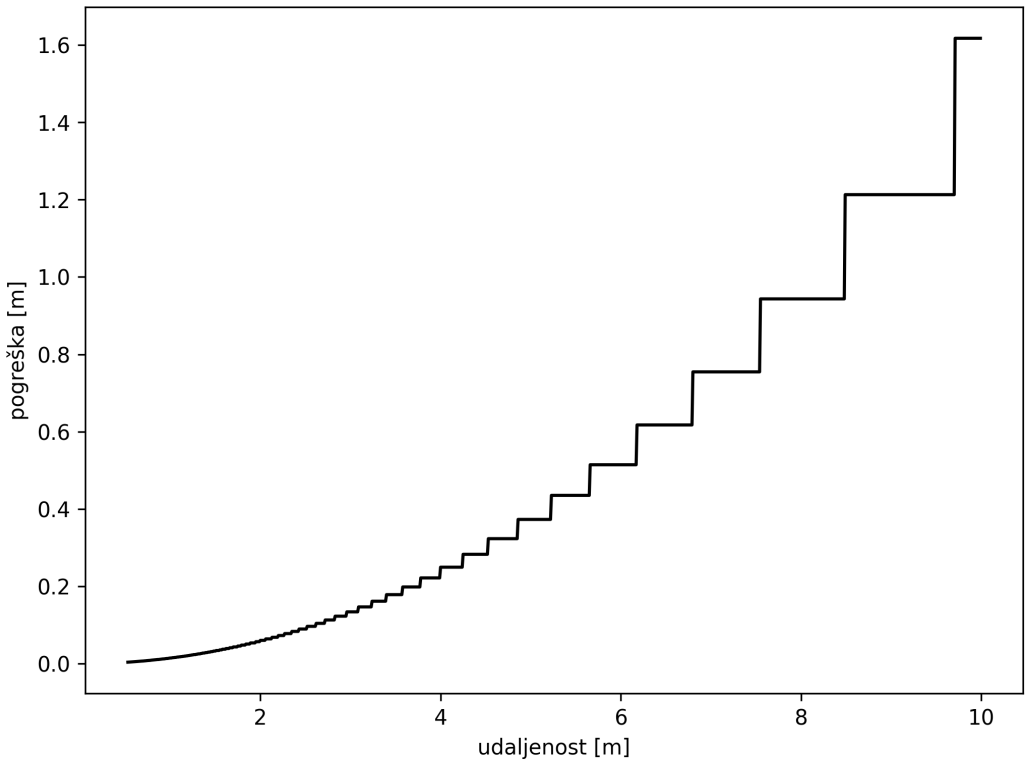
Ova formula proširi se izračunom vrijednosti dispariteta dobivenim formulom 4.14 iz udaljenosti voksela $d_v(v_o)$:

$$error_d(v_o) = \frac{T_x \cdot f}{\left\lfloor \frac{T_x \cdot f}{d_v(v_o)} \right\rfloor} - \frac{T_x \cdot f}{\left\lfloor \frac{T_x \cdot f}{d_v(v_o)} + 1 \right\rfloor}. \tag{4.16}$$

Vrijednost funkcije pogreške za stereo kameru u ovisnosti o udaljenosti od osjetila prikazana je na slici 4.7.

Kako je i ranije navedeno, u izračunu vrijednosti funkcije vidljivosti traži se da njezina domena bude $v_v(v_o) \in [0, 1]$. Pretpostavka je da vrijednost jedan predstavlja potpunu vidljivost nekog voksela, dok vrijednost nula opisuje naveden voksel kao nevidljiv. Ovaj princip potrebno je preslikati na model stereo kamere.

U predloženom modelu, ako je udaljenost dviju uzastopnih razina dispariteta opisana

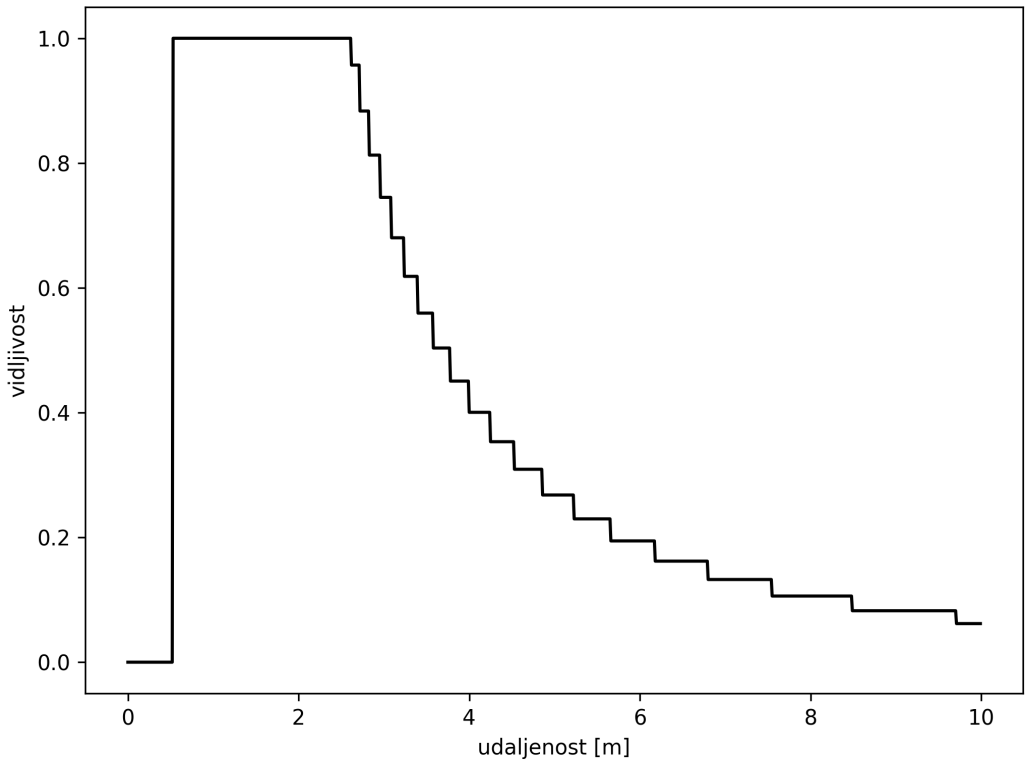


Slika 4.7: Primjer funkcije pogreške $error_d(v_o)$ u ovisnosti o udaljenosti za stereo kamere

kroz funkciju pogreške $error_d(v_o)$ manja od zadane najmanje veličine objekta kojeg se želi detektirati $error_{thr}$, pretpostavlja se da je voksel na toj udaljenosti potpuno vidljiv. U tom slučaju, objekt će biti moguće raspoznati od ostatka scene u kojoj se nalazi. Nadalje, kako udaljenost od osjetila raste, raste i pogreška nastala zbog udaljenosti uzastopnih razina dispariteta. Porastom pogreške, smanjuje se sposobnost detekcije objekta od ostatka scene te vidljivost opada obrnuto proporcionalno vrijednosti pogreške:

$$v_d(v_o) = \begin{cases} 1, & \text{akko } error_d(v_o) < error_{thr}, \\ \dfrac{error_{thr}}{error_d(v_o)}, & \text{inače.} \end{cases} \tag{4.17}$$

Funkcija vidljivosti u ovisnosti o udaljenosti $v_d(v_o)$ za stereo kameru prikazana je na slici 4.8. Na navedenoj slici, za vrijednosti dispariteta veće od d_{max}, izvodi se stereo usporedba te stereo kamera uobičajeno ne vidi točke scene blizu ravnine u kojoj se kamere nalaze.

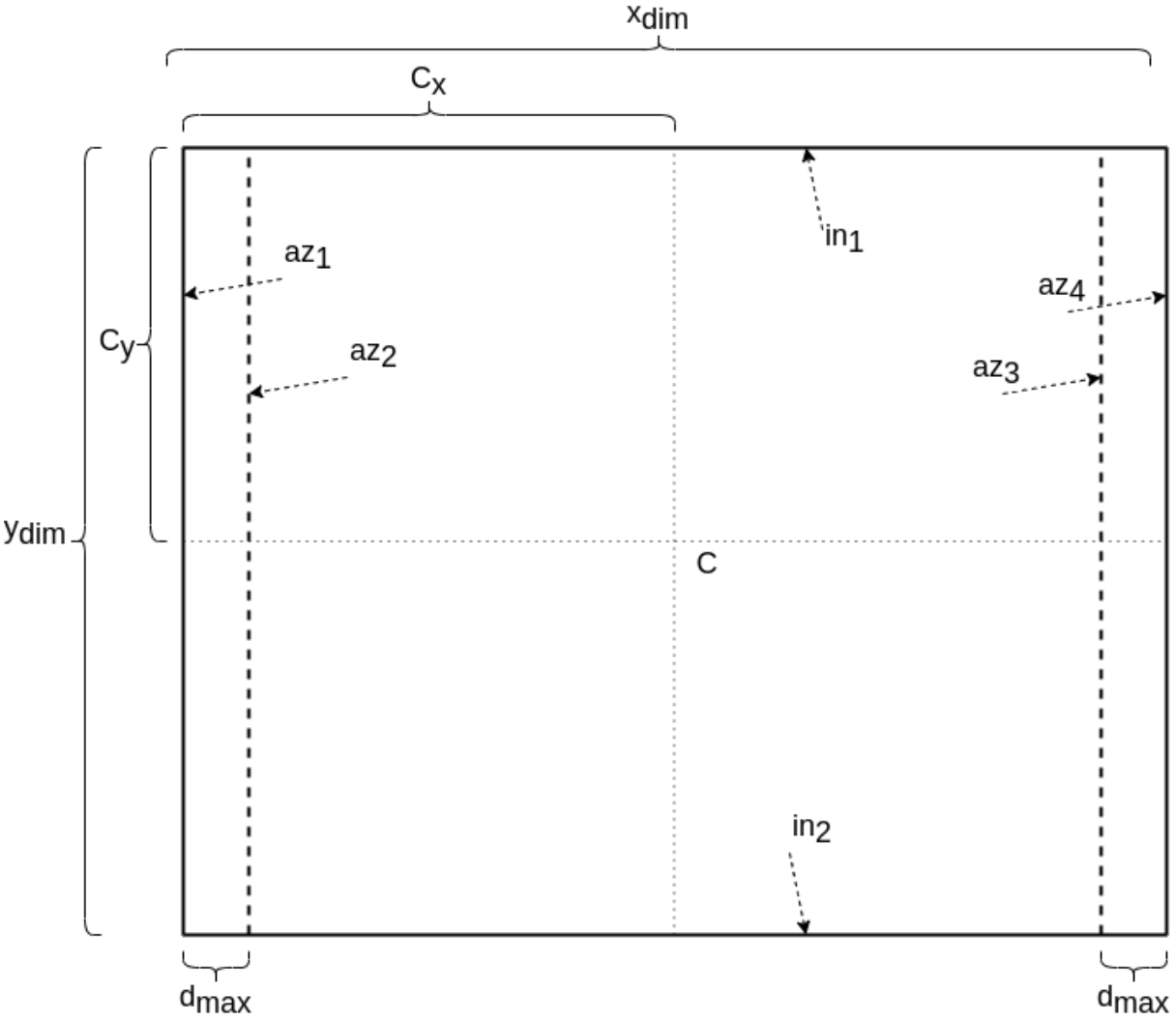


Slika 4.8: Primjer funkcije vidljivosti u ovisnosti o udaljenosti $v_d(v_o)$ za stereo kameru

Funkcije vidljivosti ovisne o kutevima azimuta i inklinacije ovise o dvije glavne značajke stereo kamere: vidnom polju te o samom algoritmu stereo usporedbe, odnosno pretrage za sličnim pikselima. Oba vidna polja kamere, vertikalno i horizontalno, utječu na domenu vidljivog dijela funkcije vidljivosti odgovarajućeg kuta. O odabranom algoritmu stereo usporedbe ovise ostale značajke funkcije vidljivosti, posebice nagib na prijelazima između potpuno vidljivog i nevidljivog dijela funkcije.

Nekoliko je značajnih vrijednosti u dubinskoj slici koja je nastala kao rezultat postupka stereo usporedbe, a koje se koriste pri izračunu funkcija vidljivosti ovisnih o kutevima, kako je prikazano na slici 4.9. Pune podebljane linije predstavljaju okvir, rubove, slike širine x_{dim} i visine y_{dim}. Dvije tanke istočkane linije presijecaju se u centralnoj točki C. Centralna točka C odmaknuta je od lijevog ruba za c_x te od gornjeg ruba za c_y. Dvije tanke iscrtkane linije odmaknute su od lijevog i desnog ruba za vrijednost maksimalnog dispariteta d_{max}.

Na slici 4.9 strelice označene s az_1, az_2, az_3 i az_4 pokazuju na vertikalne granice koje



Slika 4.9: Prikaz vertikalnih i horizontalnih granica značajnih za izračun funkcija vidljivosti ovisnih o kutevima

se koriste se pri izračunu funkcije vidljivosti u ovisnosti o azimutu. Linije označene s az_1 i az_4 vanjski su rubovi same slike te su njihove vrijednosti udaljenost te linije od centralne točke C:

$$az_1 = 0 - c_x \tag{4.18}$$

i

$$az_4 = x_{dim} - c_x. \tag{4.19}$$

Kutevi na tim granicama odgovaraju rubnim vrijednostima unutar kojih se odgovarajuće točke u prostoru scene, vokseli, smatraju vidljivima. Ako je kut između osjetila i voksela veći od navedenog, voksel nije unutar horizontalnog vidnog polja te posljedično

nije vidljiv. Vrijednosti kuteva na tim pozicijama su:

$$\angle az_1 = \arctan 2(T_x \cdot az_1, T_x \cdot f), \tag{4.20}$$

te

$$\angle az_4 = \arctan 2(T_x \cdot az_4, T_x \cdot f). \tag{4.21}$$

Drugi skup strelica, az_2 i az_3, koje pokazuju na iscrtkane vertikalne granice, predstavljaju vrijednosti udaljene za d_{max} od rubova prema centralnoj točki C:

$$az_2 = d_{max} - c_x, \tag{4.22}$$

od lijevog ruba te:

$$az_3 = x_{dim} - d_{max} - c_x, \tag{4.23}$$

od desnog ruba.

Vrijednosti kuteva izračunavaju se isto kao i u prethodna dva slučaja te iznose:

$$\angle az_2 = \arctan 2(T_x \cdot az_2, T_x \cdot f) \tag{4.24}$$

i:

$$\angle az_3 = \arctan 2(T_x \cdot az_3, T_x \cdot f). \tag{4.25}$$

Svi vokseli unutar ovih dviju vrijednosti kuteva potpuno su vidljivi. Za voksele između vanjskih rubnih kuteva $\angle az_1$ i $\angle az_4$ te unutarnjih rubnih kuteva $\angle az_2$ i $\angle az_3$, vidljivost je definirana kao omjer udaljenosti od unutarnje granice i udaljenosti unutarnje i vanjske granice d_{max}:

$$v_\varphi(v_o) = \begin{cases} \dfrac{\varphi(v_o) - \angle az_1}{\angle az_2 - \angle az_1}, & \text{akko } \angle az_1 \leq \varphi(v_o) < \angle az_2, \\ 1, & \text{akko } \angle az_2 \leq \varphi(v_o) < \angle az_3, \\ \dfrac{\varphi(v_o) - \angle az_4}{\angle az_3 - \angle az_4}, & \text{akko } \angle az_3 \leq \varphi(v_o) < \angle az_4, \\ 0, & \text{inače.} \end{cases} \tag{4.26}$$

Ovaj izračun proizlazi iz samog procesa postupka stereo usporedbe i manjeg prostora

pretrage za sličnim pikselima u blizini rubova. Ovisnost funkcije vidljivosti o kutu azimuta $v_\varphi(v_o)$ prikazana je na slici 4.10.

Slika 4.10: Primjer funkcije vidljivosti u ovisnosti o kutu azimuta $v_\varphi(v_o)$ za stereo kameru

Dvije strelice označene s in_1 i in_2 na slici 4.9, pokazuju na horizontalne rubove te će se kutevi, izračunati za te dvije pozicije po y koordinati, koristiti pri izračunu funkcije vidljivosti u ovisnosti o inklinaciji. Navedene dvije vrijednosti predstavljaju udaljenosti gornjeg i donjeg horizontalnog ruba slike od centralne točke slike. Udaljenost linije označene sa in_1 na vrhu slike od centralne točke C je:

$$in_1 = 0 - c_y, \tag{4.27}$$

a odgovarajući kut inklinacije točke u prostoru scene na toj horizontali je:

$$\angle in_1 = \arctan 2(T_x \cdot in_1, T_x \cdot f). \tag{4.28}$$

Udaljenost druge linije označene sa in_2, oznake koja se nalazi na dnu slike, od centralne

točke C iznosi:

$$in_2 = y_{dim} - c_y, \tag{4.29}$$

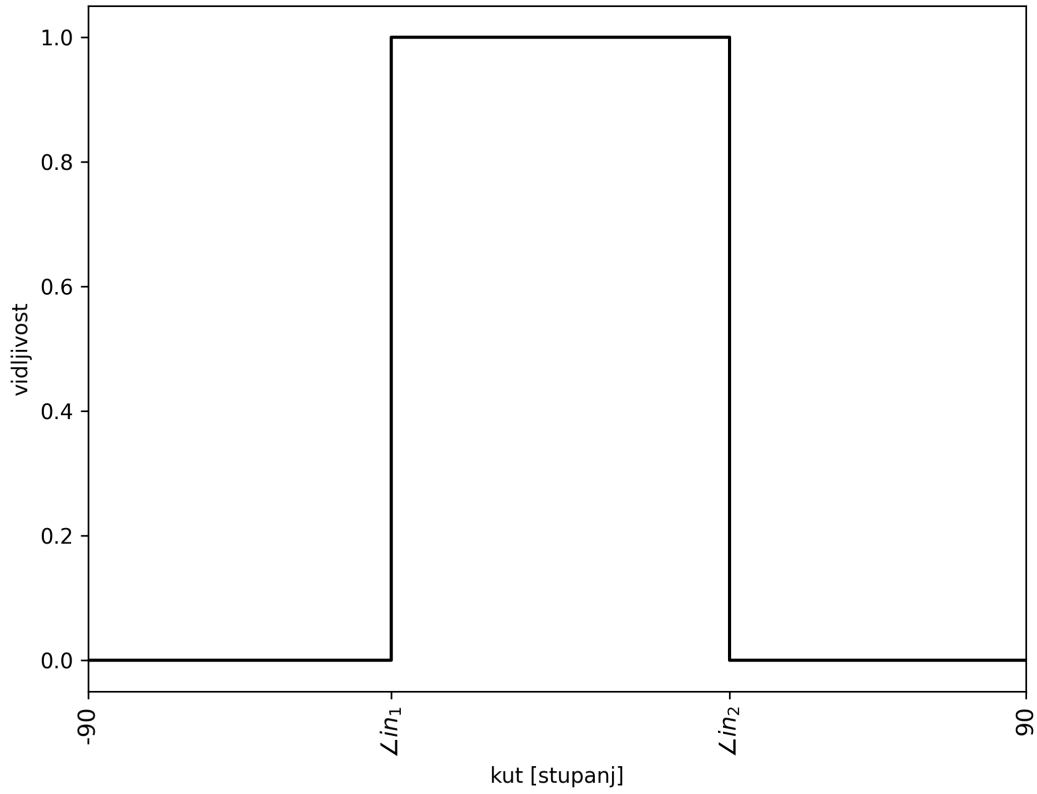
dok je odgovarajući kut inklinacije točke u prostoru scene:

$$\angle in_2 = \arctan 2(T_x \cdot in_2, T_x \cdot f). \tag{4.30}$$

Kako se u postupku stereo usporedbe radi usporedba sličnosti samo unutar istog retka u slici, funkcija u ovisnosti o kutu inklinacije ima samo dva stanja, nula i jedan. Kao i pri izračunu funkcije vidljivosti u ovisnosti o kutu azimuta, vokseli čiji se kut nalazi unutar vertikalnog vidnog polja, su potpuno vidljivi, dok su oni izvan nevidljivi. Izračunati kutevi $\angle in_1$ i $\angle in_2$ granične su vrijednosti vertikalnog vidnog polja te se koriste pri izračunu funkcije vidljivosti u ovisnosti o kutu inklinacije:

$$v_\theta(v_o) = \begin{cases} 1, & \text{akko } in_1 \leq \theta(v_o) < in_2, \\ 0, & \text{inače.} \end{cases} \tag{4.31}$$

Na slici 4.11 prikazana je rezultantna funkcija vidljivosti u ovisnosti o kutu inklinacije. Vidljivo je da su rubovi, za razliku od funkcije vidljivosti o kutu azimuta, vertikalni te postoji trenutni prijelaz između dvije vrijednosti. Kod dvodimenzionalnih modela prostora, vrijednost funkcije vidljivosti ovisne o kutu inklinacije iznosi $v_\theta(v_o) = 1$.



Slika 4.11: Primjer funkcije vidljivosti u ovisnosti o kutu inklinacije $v_\theta(v_o)$ za stereo kameru

5. Poglavlje

MODEL RAZMJEŠTAJA OSJETILA ZA POKRIVANJE ZATVORENOG PROSTORA

Treći doprinos ove doktorske disertacije model je razmještaja osjetila za pokrivanje zatvorenoga prostora. Predloženi model razmještaja osjetila uključuje prethodno opisane modele prostora i osjetila te optimizacijsku funkciju koja se koristi pri izračunu pokrivenosti prostora osjetilnim sposobnostima osjetilima, odnosno ukratko osjetilima. Koristeći predloženi model razmještaja osjetila moguće je razmjestiti osjetila u dvodimenzionalnom i trodimenzionalnom zatvorenom prostoru.

Razmještaj je konfiguracija pozicija i orijentacija m zadanih osjetila iz skupa osjetila S u slobodnom prostoru V opisanim s n voksela. Ovisno o primjeni, korisnik definira broj i vrstu osjetila koja se razmještaju.

5.1. Optimizacijska funkcija razmještaja osjetila

U postupku razmještanja osjetila potrebno je izračunati združeni utjecaj njihovog razmještaja, pozicija i orijentacija, na pokrivenost prostora u kojem se nalaze. Funkcija kojom se vrednuje razmještaj osjetila vodeći računa o topografskim značajkama prostora naziva se optimizacijska funkcija razmještaja osjetila ili skraćeno samo optimizacijska funkcija.

Optimizacijska funkcija temelji se na izračunu funkcije vidljivosti $v_{vs}(v_i, s_j)$ za sve parove voksela u prostoru v_i i osjetila s_j. Vrijednost funkcije vidljivosti jednog para $v_{vs}(v_i, s_j)$ ovisi o modelu osjetila s_j te odnosu, udaljenosti i kutevima, između voksela v_i i osjetila s_j. Prije samog izračuna funkcije vidljivosti $v_{vs}(v_i, s_j)$ provjerava se postojanje optičke vidljivosti između voksela i osjetila, odnosno, provjerava se ako postoji prepreka koja prekida pravac između voksela v_i i osjetila s_j. Ako ne postoji optička vidljivost, odnosno, postoji prepreka na pravcu koji povezuje voksel i osjetilo, tada odredišni voksel v_i nije vidljiv iz osjetila s_j te je vidljivost za taj par $v_{vs}(v_i, s_j)$ jednaka nuli.

Ako optička vidljivost postoji, provede se transformacija pozicije svakog voksela v_i iz globalnog koordinatnog sustava prostora u pozicije voksela v_o u lokalnom koordinatnom sustavu osjetila s_j radi izračuna vidljivosti koristeći model osjetilne sposobnosti osjetila s_j. Za tu transformaciju koriste se značajke osjetila, odnosno odgovarajuće matrice translacije i rotacije. Svako osjetilo s_j opisano je koordinatama s_{j_x}, s_{j_y} i s_{j_z} svoje pozicije u globalnom koordinatnom sustavu prostora. Matrica translacije T sadrži podatke o poziciji osjetila u prostoru:

$$T = \begin{bmatrix} 1 & 0 & 0 & -s_{j_x} \\ 0 & 1 & 0 & -s_{j_y} \\ 0 & 0 & 1 & -s_{j_z} \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{5.1}$$

Usmjerena osjetila imaju i podatke o svojoj orijentaciji. Orijentacija osjetila opisana je koristeći tri kuta: kut uvijanja (eng. roll) s_{j_R} oko x osi, kut nagiba (eng. pitch) s_{j_P} oko y osi te kut zaošijanja (eng. yaw) s_{j_Y} oko z osi.

Rotacija oko x osi opisana je matricom rotacije R_x:

$$R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos(s_{j_R}) & -sin(s_{j_R}) & 0 \\ 0 & sin(s_{j_R}) & cos(s_{j_R}) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{5.2}$$

rotacija oko y osi matricom rotacije R_y:

$$R_y = \begin{bmatrix} cos(s_{j_P}) & 0 & sin(s_{j_P}) & 0 \\ 0 & 1 & 0 & 0 \\ -sin(s_{j_P}) & 0 & cos(s_{j_P}) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{5.3}$$

a matricom rotacije R_z opisana je rotacija oko z osi:

$$R_z = \begin{bmatrix} cos(s_{j_Y}) & -sin(s_{j_Y}) & 0 & 0 \\ sin(s_{j_Y}) & cos(s_{j_Y}) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{5.4}$$

Matričnim množenjem pozicije voksela v_i u globalnom koordinatnom sustavu prostora s matricama translacije i rotacije, dobije se pozicija voksela v_o u koordinatnom sustavu odgovarajućeg osjetila s_j:

$$\begin{bmatrix} v_{o_x} \\ v_{o_y} \\ v_{o_z} \\ 1 \end{bmatrix} = R_z \cdot R_y \cdot R_x \cdot T \cdot \begin{bmatrix} v_{i_x} \\ v_{i_y} \\ v_{i_z} \\ 1 \end{bmatrix}. \tag{5.5}$$

Izračunate vrijednosti vidljivosti $v_v(v_o)$ za svaki par voksela v_i i osjetila s_j u lokalnom koordinatnom sustavu osjetila s_j mapiraju se u funkciju vidljivosti $v_{vs}(v_i, s_j)$ u globalnom koordinatnom sustavu prostora:

$$f : v_v(v_o) \mapsto v_{vs}(v_i, s_j) \quad \forall s_j \in S \tag{5.6}$$

U daljnjim će se izračunima koristiti vrijednost funkcije gubitka između svakog od voksela u prostoru i osjetila. Funkcija gubitka svakog od voksela u prostoru i osjetila predstavlja komplement vrijednosti funkcije vidljivosti $v_{vs}(v_i, s_j)$:

$$l_{vs}(v_i, s_j) = 1 - v_{vs}(v_i, s_j). \tag{5.7}$$

Funkcija gubitka $l_{vs}(v_i, s_j)$ ima istu domenu kao i funkcija vidljivosti $v_{vs}(v_i, s_j)$:

$$l_{vs}(v_i, s_j) \in [0, 1]\,. \tag{5.8}$$

Kako jedan voksel u prostoru može biti vidljiv iz više osjetila u istom trenutku, gubitak svakog voksela $L_v(v_i, S)$ definira se kao umnožak gubitaka između voksela v_i i svakog od m osjetila s_j iz skupa osjetila S:

$$L_v(v_i, S) = \prod_{j=1}^{m} l_{vs}(v_i, s_j). \tag{5.9}$$

Vidljivost svakog voksela u prostoru $V_v(v_i, S)$ komplement je njegovog gubitka $L_v(v_i, S)$:

$$V_v(v_i, S) = 1 - L_v(v_i, S). \tag{5.10}$$

Domene i ovih funkcija su:

$$L_v(v_i, S),\, V_v(v_i, S) \in [0, 1]\,. \tag{5.11}$$

Osnovni razlog korištenja funkcije gubitka između svakog od voksela u prostoru i osjetila $l_{vs}(v_i, s_j)$ je u osiguravanju odgovarajućeg utjecaja vidljivosti svakog od osjetila $v_{vs}(v_i, s_j)$ na vidljivost jednog voksela $V_v(v_i, S)$. Umnoškom gubitaka $l_{vs}(v_i, s_j)$ između svakog od osjetila s_j koje vidi zadani voksel v_i vrijednost funkcije gubitka voksela $L_v(v_i, S)$ bit će manja od svakog pojedinačnog gubitka.

Optimizacijska funkcija u ovom rješenju je funkcija gubitka $L(V, S)$, funkcija čiju se vrijednosti pri postupku optimizacije minimizira. Ta optimizacijska funkcija proizlazi iz odabrane metrike pokrivenosti. Metrika pokrivenosti korištena u predloženom modelu je metrika pokrivenosti područja uz vjerojatnosne osjetilne sposobnosti, formula 1.2. Ova metrika sve voksele u prostoru smatra jednako važnima te zbog toga funkcija gubitka $L(V, S)$ predstavlja aritmetičku sredinu gubitaka $L_v(v_i, S)$ za svaki od n voksela v_i u slobodnom prostoru V:

$$L(V, S) = \frac{1}{n} \sum_{i=1}^{n} L_v(v_i, S). \tag{5.12}$$

Iako se za predloženo rješenje koristi minimizacijska funkcija gubitka koja kao cilj

ima smanjiti vrijednost gubitka $L(V, S)$ pri prikazu rezultata u kasnijim poglavljima, ali i za dio problema koji zahtijevaju maksimizacijsku funkciju, može se iskoristiti vrijednost pokrivenosti $C(V, S)$:

$$C(V, S) = 1 - L(V, S). \tag{5.13}$$

Domene i funkcije gubitka $L(V, S)$ i funkcije pokrivenosti $C(V, S)$ su:

$$L(V, S),\, C(V, S) \in [0, 1]\,. \tag{5.14}$$

Kao što je ranije navedeno, predložena je funkcija gubitka $L(V, S)$, kod koje je pri optimizaciji cilj minimizacija njene vrijednosti. Također, predložena je i funkcija pokrivenosti $C(V, S)$ koja se može koristiti kod optimizacijskih postupaka temeljenih na maksimizaciji vrijednosti.

Razmještaj osjetila opisan predloženim funkcijama gubitka $L(V, S)$ i pokrivenosti $C(V, S)$ je jednociljni (eng. single-objective) problem, odnosno optimizira se vrijednost samo jedne zavisne varijable. Razmještaj osjetila je i kontinuiran (eng. continuous) problem, točnije nezavisne varijable koje se koriste pri optimizaciji mogu poprimiti vrijednosti iz domene realnih brojeva.

Za optimizaciju problema razmještaja osjetila koristeći predložene optimizacijske funkcije može se iskoristiti svaki optimizacijski algoritam pogodan za nelinearnu (eng. nonlinear) i ograničenu (eng. constrained) optimizaciju. Nelinearnost optimizacijskog problema proizlazi iz ovisnosti rezultata optimizacijske funkcije za različite prostore, različitih vrsta osjetilnih sposobnosti osjetila te broja osjetila čiji se razmještaj optimizira. Optimizacija je ograničena unutar domene prostora, odnosno moguća rješenja pozicija svakog osjetila su iz domene realnoj brojeva, ali se moraju nalaziti unutar slobodnog dijela modela prostora. Slično vrijedi i za kuteve orijentacije čiji su limiti u domeni $[-180, 180]$ za kut zaošijanja te $[-90, 90]$ za kuteve nagiba i uvijanja. Dodatno, predložene optimizacijske funkcije nisu derivabilne u cijeloj svojoj domeni tako da optimizacijski algoritam mora biti pogodan za optimizaciju koja ne koristi derivacije funkcije (eng. derivative-free).

6. Poglavlje

SIMULACIJSKO OKRUŽENJE

Modeli prostora, osjetilnih sposobnosti osjetila i razmještaja osjetila opisani u prethodnim poglavljima razvijeni su u simulacijskom okruženju. Osnovna svrha simulacijskog okruženja je provjera modela razmještaja osjetila u zatvorenom prostoru. Omogućuje se i ispitivanje različitih optimizacijskih algoritama za problem razmještaja svesmjernih i usmjerenih osjetila u raznim modelima prostora.

Simulacijsko okruženje razvijeno je u programskom jeziku *python*, verzije 3.6.8. Osnovne korištene knjižnice funkcija su *numpy* za većinu numeričkih izračuna, *shapely* za rad s poligonima te *NiaPy* koja sadrži implementacije optimizacijskih algoritama.

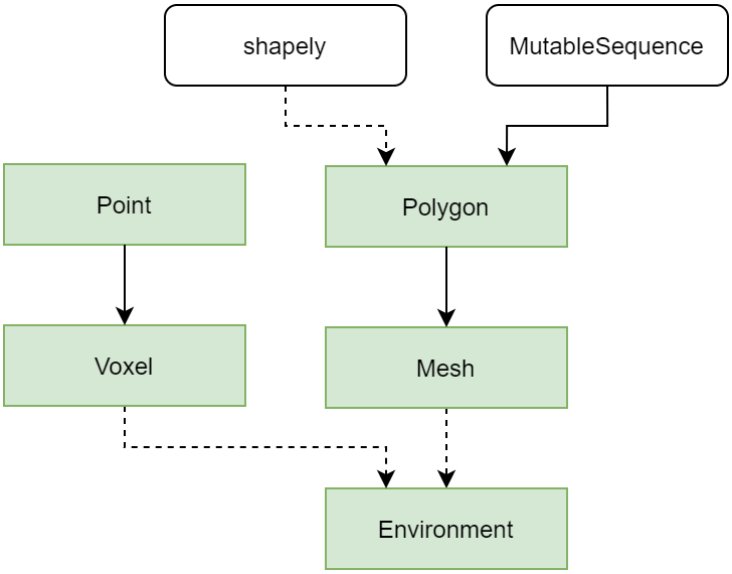
6.1. Simulacija prostora

Prvi segment simulacijskog okruženja odnosi se na model prostora. Model prostora opisan je u simulacijskom okruženju kroz nekoliko klasa. Dijagram klasa kojima su elementi prostora opisani prikazan je na slici 6.1. Klasa *Point*, izvorišni kod A.1, sadrži definiciju svake točke prostora, odnosno njezine koordinate. Svaka točka prostora proširuje se podatcima o vidljivosti $V_v(v_i, S)$ i gubitku $L_v(v_i, S)$ u klasi *Voxel*, izvorišni kod A.2.

Klasa *Polygon* nasljeđuje klasu *MutableSequence* i omogućava izradu poligona, izvorišni kod A.3. Klasa *Polygon* sadrži operacije za rad s točkama, linijama i poligonima. Navedene operacije koriste objekte i funkcije iz programskog paketa *shapely*. Ova klasa i u njoj zapisani podatci koriste se za definiciju baznog poligona mreže prostora. Mreže prostora su opisane klasom *Mesh*, izvorišni kod A.4, koja nasljeđuje klasu *Polygon*. U

odnosu na klasu *Polygon* klasa *Mesh* sadrži i podatke o visinama zadane mreže kao i funkcije koje se koriste pri provjeri optičke vidljivosti.

Klasa *Environment* kojom je prostor opisan koristi ranije opisane klase *Voxel* i *Mesh*, izvorišni kod A.5. Kako je navedeno u poglavlju 3., prostor se sastoji od mreža slobodnog prostora i prepreka. U klasi *Environment* dostupne su funkcije za stvaranje navedenih mreža prostora, kao i funkcija za izradu prostora postupkom rasterizacije. Također, dostupne su i funkcije za provjeru optičke vidljivosti i ispravnosti točke. Obje navedene funkcije oslanjaju se na istovrsne funkcije u klasi *Mesh* i posljedično u klasi *Polygon*.



Slika 6.1: Dijagram klasa opisa prostora

Svaki prostor opisan je u datoteci s postavkama u *Json* formatu [77]. U izvorišnom kodu 6.1 dan je primjer opisa prostora u obliku slova „L" s preprekom, prikazanog na slici 3.2. U prvoj liniji definicije svakog prostora definira se ključ, primjerice *„env8"*, koji služi za povezivanje opisanog prostora u ostatku datoteke s postavkama. Unutar bloka pridruženog tom ključu nalazi se ime prostora te njegov opis. Pri opisu prostora koriste se opisi mreža slobodnog prostora (*„positivemeshes"*) i prepreka (*„negativemeshes"*). Svaka mreža opisana je točkama svog baznog poligona te donjom i gornjom visinom. U definiciji svakog prostora mora se nalaziti barem jedna mreža slobodnog prostora. Gornjih granica za broj mreža slobodnog prostora i prepreka nema.

Izvorišni kod 6.1: Primjer opisa prostora u *Json* formatu

```
1 "environments": {
2     ...,
```

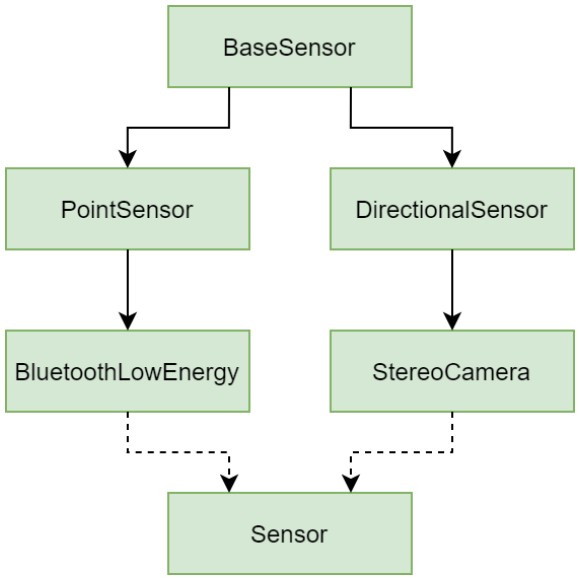
```
    "env8": {
        "name": "env8",
        "definition": {
            "positivemeshes": [
                {
                    "polygon": [[0, 0], [5, 0], [5, 5], [3, 5], [3, 2],
                       [0, 2], [0, 0]],
                    "bottom": 0,
                    "top": 5
                }
            ],
            "negativemeshes": [
                {
                    "polygon": [[3, 1], [4, 1], [4, 2], [3, 2], [3, 1]],
                    "bottom": 0,
                    "top": 2
                }
            ]
        }
    },
    ...
}
```

6.2. Simulacija osjetilnih sposobnosti osjetila

Drugi segment simulacijskog okruženja opisuje modele osjetilnih sposobnosti osjetila. Klasa *BaseSensor*, izvorišni kod A.6, sadrži osnovne funkcije za izračun udaljenosti i kuteva između dviju točaka. Klase *PointSensor*, izvorišni kod A.7, i *DirectionalSensor*, izvorišni kod A.8, sadrže definicije osnovnih funkcija za rad s navedenim vrstama osjetila te funkcije za postavljanje pozicija i orijentacija samih osjetila.

Klasa *BluetoothLowEnergy*, izvorišni kod A.9, sadrži definicije potrebne za izračun funkcije vidljivosti u ovisnosti o udaljenosti za predloženi model osjetilne sposobnosti svesmjernog osjetila. U klasi *StereoCamera*, izvorišni kod A.10, opisana je osjetilna sposobnost usmjerenog osjetila, odnosno definirane su funkcije za izračune odgovarajućih vidljivosti.

Kako bi se omogućilo instanciranje objekta odgovarajućeg osjetila uz zadovoljavanje preduvjeta učahurivanja (eng. encapsulation) izvorišnog koda, izrađena je klasa *Sensor*, izvorišni kod A.11. Navedena klasa instancira odgovarajući objekt temeljem imena osjetila kojeg prima kao argument. Dijagram opisanih klasa dan je na slici 6.2.



Slika 6.2: Dijagram klasa opisa osjetilnih sposobnosti osjetila

Značajke osjetila su također definirane u *Json* datoteci s postavkama, izvorišni kod 6.2. Svako osjetilo opisano je svojim ključem, primjerice *„ble”* i *„sc”*. Unutar bloka definicije svakog osjetila nalazi se ime osjetila te njegove značajke.

Izvorišni kod 6.2: Primjer zadavanja značajki osjetila u *Json* formatu

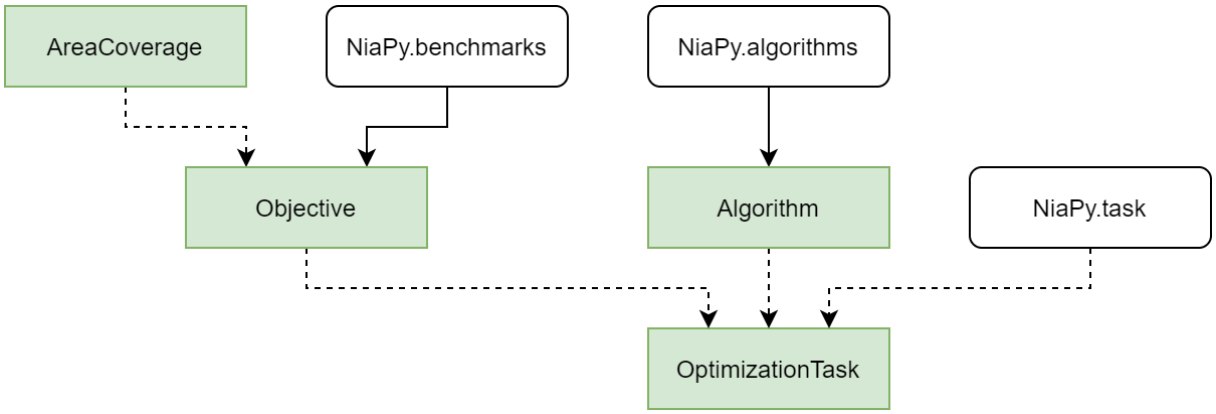
```
"sensors": {
    "ble": {
        "name": "BluetoothLowEnergy",
        "params": {
            "rssi1m": -75,
            "noise": -85
        }
    },
    "sc": {
        "name": "StereoCamera",
        "params": {
            "b": 0.12,
            "fx": 565.9872,
            "x_dim": 1280,
```

```
            "y_dim": 720,
            "nd": 128,
            "cx": 702.4413,
            "cy": 351.1992,
            "threshold": 0.1
        }
    }
}
```

6.3. Optimizacijski zadatak

Optimizacijski zadatak opisan klasom *OptimizationTask*, izvorišni kod A.12, povezuje optimizacijski algoritam s optimizacijskom funkcijom temeljenom na metrici pokrivenosti područja. Kako je ranije navedeno, u ovom je simulacijskom okruženju korištena knjižnica funkcija *NiaPy*.

Klasa *OptimizationTask* koristi funkcije iz *NiaPy* knjižnice funkcija za opis vrste optimizacije te uvjeta za prekid optimizacije. Vrsta optimizacije, kako je to navedeno u poglavlju 5., je minimizacija funkcije gubitka. Optimizacija se prekida kad se izvrši odgovarajući broj izvršavanja optimizacijske funkcije, što će biti detaljnije opisano u poglavlju 7.3.. Na slici 6.3 dan je prikaz dijagrama klasa opisa optimizacijskog zadatka.



Slika 6.3: Dijagram klasa opisa optimizacijskog zadatka

Klasa *Algorithm*, izvorišni kod A.13, služi za učahurivanje optimizacijskih algoritama dostupnih u knjižnici funkcija *NiaPy*. Značajke algoritama zadane su u *Json* datoteci s postavkama, izvorišni kod 6.3. Svaki algoritam opisan je svojim ključem, primjerice

„pso”, koji se koristi za povezivanje opisanog algoritma u ostatku datoteke s postavkama. Unutar bloka nalaze se ime algoritma te značajke koje se koriste pri optimizaciji.

Izvorišni kod 6.3: Primjer zadavanja značajki algoritama u *Json* formatu

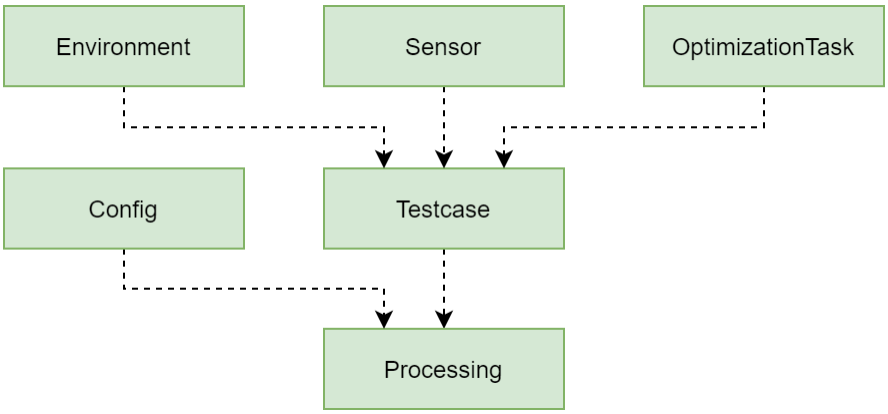
```
"algorithms": {
    ...,
    "pso": {
        "name": "ParticleSwarmOptimization",
        "params": {
            "NP": 10,
            "C1": 1,
            "C2": 1,
            "w": 0.7
        }
    },
    ...
}
```

Klasa *Objective*, izvorišni kod A.14, nasljeđuje klasu *Benchmark* iz knjižnice funkcija *NiaPy* te u odgovarajućem formatu definira granice optimizacije i funkciju gubitka. Funkcija gubitka $L(V, S)$ definirana je u klasi *AreaCoverage*, izvorišni kod A.15.

6.4. Ispitni slučaj i višedretvena obrada

Sva tri prethodna segmenta, prostori, osjetilne sposobnosti osjetila te optimizacijski zadatak povezani su u ispitni slučaj. Ispitni slučaj opisan je u klasi *Testcase*, izvorišni kod A.16. U ovoj klasi nalaze se funkcije koje se koriste za izradu prostora i dodavanje osjetila u prostor. Klasa *Config*, izvorišni kod A.17, pomoćna je klasa koja služi za učitavanje podataka iz *Json* datoteke s postavkama.

Klasa *Processing*, izvorišni kod A.18, koristi podatke učitane iz *Json* datoteke i temeljem njih izrađuje skup ispitnih slučajeva. Navedena datoteka, izvorišni kod 6.4, sadrži popis ključeva modela prostora u koje se razmještaju osjetila te dimenzije modela prostora i rasterizacijske parametre. Također, sadrži i popis vrsta osjetila povezanih ključevima na njihove postavke i broj osjetila koja treba razmjestiti u prostor. Zadan je i popis ključeva algoritama kojima će se izvršiti optimizacija zadane optimizacijske funkcije. Uz



Slika 6.4: Sažeti dijagram klasa simulacijskog okruženja

navedeno, zadani su i dodatni podatci poput broja evaluacija po nezavisnoj varijabli te broja ponavljanja izvođenja svakog ispitnog slučaja.

Izvorišni kod 6.4: Primjer zadavanja postavki ispitnih slučajeva u *Json* formatu

```
"processing": {
    ... ,
    "environments": ["env1", "env2", "env3"],
    "dimensions": [2, 3],
    "rasterization": [0.1, 0.5, 1],
    "sensors": [
        {
            "type": "ble",
            "numbers": [1, 2, 3]
        },
        {
            "type": "sc",
            "numbers": [1, 2, 3]
        }
    ],
    "algorithms": ["abc", "fwa", "pso", "hca", "nmm", "sa"],
    "evals_per_dimension": [100],
    "objectives": ["area"],
    "iterations": 100
}
```

U klasi *Processing* nalaze se i funkcije za mjerenje vremena izvođenja te pohranu rezultata svakog ispitnog slučaja. Pri izvođenju se koristi višedretveni rad (eng. multi-

threading), odnosno kreira se skup dretvi (eng. thread pool) koji izvršava svaki od ispitnih slučajeva. Broj dretvi ovisi o okruženju u kojem se izvršava simulacijsko okruženje. Ako se simulacijsko okruženje izvršava na sustavu koji koristi Slurm Workload Manager [78], primjerice superračunalo „Bura" Sveučilišta u Rijeci [79], tada broj dretvi ovisi o broju dodijeljenih čvorova te broju jezgri na svakom od njih. U slučaju da se simulacijsko okruženje izvršava na nekom drugom sustavu, broj jezgri je potrebno zadati u *Json* datoteci s podatcima.

7. Poglavlje

POSTAVKE EKSPERIMENATA

Kako bi se ispitala sposobnost procjene pokrivenosti područja upotrebom predloženog modela razmještaja osjetila, ali i potvrdila mogućnost izrade modela prostora i osjetila temeljem predloženih generičkih modela, potrebno je stvoriti širok spektar različitih ispitnih slučajeva. Također, temeljem usporedbe više optimizacijskih algoritama za problem razmještaja osjetila omogućuje se odabir odgovarajućeg optimizacijskog algoritma koji odgovara potrebama i mogućnostima korisnika. Cilj eksperimenata je i provjera izračuna optimizacijske funkcije predloženog modela razmještaja osjetila koristeći stohastičke algoritme.

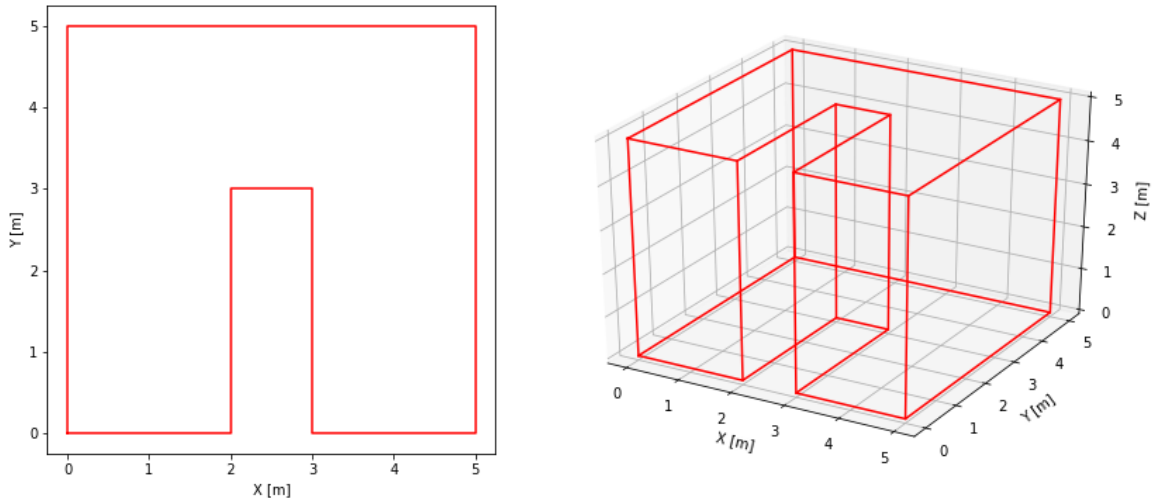
Različiti ispitni slučajevi dobiveni su variranjem korištenih prostora, njihovih dvodimenzionalnih i trodimenzionalnih modela i rasterizacijskih vrijednosti te vrsta i broja osjetila koji se razmještaju u prostoru. Ne postoje ograničenja na topografiju prostora, rasterizacijsku vrijednost niti na broj u vrstu osjetila. Iako je moguće razmjestiti različite vrste osjetila u istom modelu prostora, u ovom eksperimentu to nije urađeno. Uparivanje različitih vrsta osjetila u heterogeni sustav osjetila ovisi o samoj primjeni i vrsti signala koju ta osjetila primaju ili odašilju.

7.1. Modeli prostora

U preliminarnom dijelu istraživanja ispitan je niz različitih dvodimenzionalnih i trodimenzionalnih modela prostora. Modeli prostori su se razlikovali u dimenzijama, obliku te broju i oblicima prepreka. U ovoj disertaciji odabrani su i opisani modeli tri različita

prostora. Ta tri prostora razlikuju se u obliku te broju i oblicima prepreka u njima. Za svaki od opisanih prostora iskorišteni su njihovi dvodimenzionalni i trodimenzionalni modeli.

Ispitni prostor 1, slika 7.1, oblika je slova „U" bez ikakvih prepreka. Zbog oblika prostora nemoguće je pokriti cijeli prostor jednim osjetilom. Primjeri ovakvih, konkavnih, prostora redoviti su u stvarnim primjenama te se u znanstvenoj literaturi [55, 57, 15] često koriste kao jedan od posebnih ispitnih slučajeva pri razmještaju osjetila. Na slici 7.1 a) dan je prikaz dvodimenzionalnog modela, a na slici 7.1 b) dan je prikaz trodimenzionalnog modela navedenog prostora.

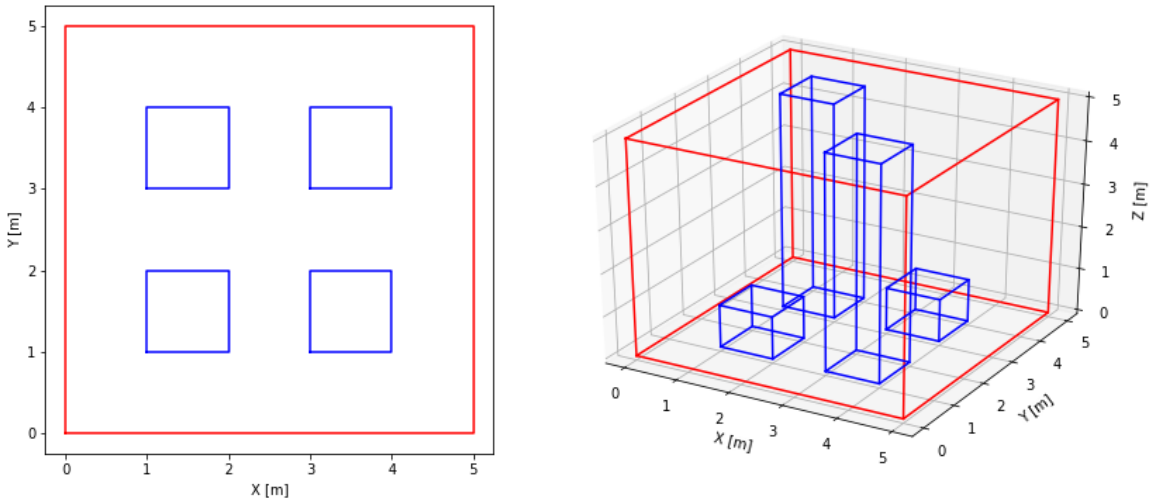


(a) (b)

Slika 7.1: Primjer dvodimenzionalnog i trodimenzionalnog modela Ispitnog prostora 1

Ispitni prostor 2, slika 7.2, primjer je simetričnog prostora. U prostoru oblika kocke postavljene su četiri prepreke, stupovi, različitih visina. Zbog simetričnosti ovoga prostora, postoje rješenja koja je moguće intuitivno prepoznati, primjerice jedno svesmjerno osjetilo se može razmjestiti u sredinu dvodimenzionalnog modela prostora i time pokriti većinu prostora. Mogu se također prepoznati i potencijalna rješenja razmještaja dva svesmjerna osjetila u trodimenzionalnom modelu prostora.

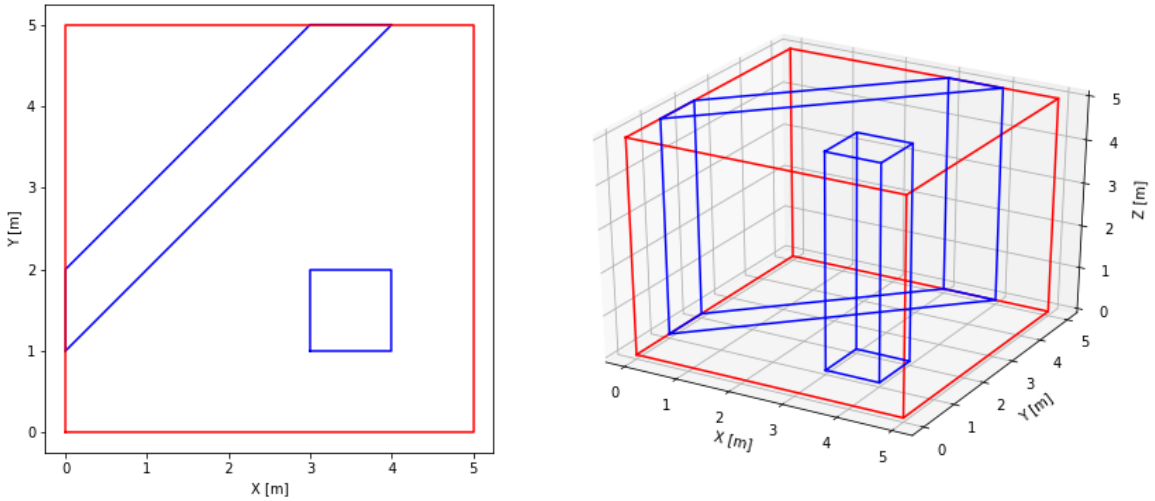
Ispitni prostor 3, slika 7.3, karakterizira pregrada koja isti dijeli na dva nepovezana dijela. Ovaj model prostora odabran je kako bi se ispitao utjecaj pregrada, odnosno prostora s više nepovezanih dijelova, na izračun pokrivenosti osjetilima. Ovaj se Ispitni prostor



(a) (b)

Slika 7.2: Primjer dvodimenzionalnog i trodimenzionalnog modela Ispitnog prostora 2

može promatrati i kao dva neovisna prostora koja je potrebno skupa pokriti zadanim brojem osjetila.



(a) (b)

Slika 7.3: Primjer dvodimenzionalnog i trodimenzionalnog modela Ispitnog prostora 3

7.1.1. Rasterizacija prostora

Modeli gore opisanih ispitnih prostora rasterizirani su s različitim rasterizacijskim vrijednostima r_v. Kako je ranije opisano, rasterizacijska vrijednost r_v definira broj voksela u prostoru. Broj voksela za dvodimenzionalne i trodimenzionalne modele sva tri ispitna prostora ovisno o rasterizacijskoj vrijednosti r_v dan je u tablici 7.1.

Tablica 7.1: Broj voksela za dvodimenzionalne (2D) i trodimenzionalne (3D) modele sva tri ispitna prostora

r_v [m]	Ispitni prostor 1		Ispitni prostor 2		Ispitni prostor 3	
	2D	3D	2D	3D	2D	3D
1	36	216	20	152	23	138
0,5	115	1265	85	1079	88	968
0,1	2331	116550	2117	115530	2099	104950

Korištenjem različitih rasterizacijskih vrijednosti varira se broj ispitnih točaka i tako omogućava analiza ostvarene pokrivenosti i vremena izvođenja s više optimizacijskih algoritama za modele koji s više ili manje detalja opisuju ispitni prostor. Kombinacijom triju ispitnih prostora, njihovih dvodimenzionalnih i trodimenzionalnih modela te tri rasterizacijske vrijednosti stvoreno je 18 različitih ispitnih slučajeva.

7.2. Modeli osjetila

Za predstavnika svesmjernih osjetila, opisanih u poglavlju 4.1., odabrana je radio-frekvencijska signalna oznaka tvrtke Estimote [80]. Navedena *Bluetooth* oznaka niske energije (eng. Bluetooth Low Energy Beacon) odašilje signal u ISM (eng. Industrial, Scientific and Medical) frekvencijskom pojasu od 2,4 GHz do 2,5 GHz. Navedeni ISM frekvencijski pojas namijenjen je industrijskim, znanstvenim te medicinskim primjenama, iz čega i proizlazi njegovo ime.

Estimote signalna oznaka periodički odašilje radio-frekvencijski signal čija nominalna jačina na udaljenosti od jednog metra iznosi $rssi_{1m} = -75\,dB$. Pretpostavljena vrijednost šuma kod koje signal više nije moguće razaznati od ostalih elektromagnetskih zračenja u prostoru iznosi $noise = -85\,dB$. Navedene vrijednosti preuzete su iz službene dokumentacije osjetila.

Za predstavnika usmjerenih osjetila odabrana je dubinska stereo kamera, specifično, ZED kamera tvrtke Stereolabs [81]. Navedena dubinska stereo kamera sastoji se od dvije kamere u istom kućištu nazivne rezolucije 2560×1440, odnosno $2K$, razmaknute približno 120 milimetara u istoj ravnini. Za ovakve stereo kamere potrebno je odrediti relativan odnos dviju kamera koristeći jedan od postupaka umjeravanja. Postupkom umjeravanja određuju se značajke sustava kamera potrebni pri rektifikaciji te Q-matrica opisana u poglavlju 4.2.

Dubinska stereo kamera čije su značajke korišteni u ovom istraživanju nosi oznaku *SN1673*. Točne značajke dobiveni postupkom umjeravanja za navedenu stereo kameru dani su u tablici 7.2. Uz značajke stereo kamere, odabrana je i najmanja veličina objekta kojeg je potrebno detektirati u sceni od $error_{thr} = 0{,}1\,m$. U navedenoj tablici dani su sve korištene vrijednosti značajki za oba osjetila.

Tablica 7.2: Značajke odabranih osjetila

Svesmjerno osjetilo (Estimote BLE oznaka)		Usmjereno osjetilo (ZED Stereo kamera)	
$rssi_{1m}$	$-75\,dB$	T_x	$0{,}12\,m$
$noise$	$-85\,dB$	f	565,99
		x_{dim}	1280
		y_{dim}	720
		d_{max}	128
		c_x	702,44
		c_y	351,20
		$error_{thr}$	$0{,}1\,m$

U ispitnim će se slučajevima razmještati i različiti broj osjetila. Ispitat će se razmještaj homogenog sustava od jednog, dva ili tri osjetila, odnosno, za svaku od dviju vrsta osjetila ispitat će se razmještaj triju kombinacija broja osjetila. Uz 18 ispitnih slučajeva dobivenih kombiniranjem različitih prostora i njihovih modela uz tri rasterizacijske vrijednosti te šest kombinacija vrsta i broja osjetila, ukupan broj različitih kombinacija ispitnih slučajeva raste na 108.

7.3. Optimizacijski algoritmi

Kako je navedeno u poglavlju 5., za određivanje razmještaja koristeći opisane optimizacijske funkcije se može iskoristiti svaki optimizacijski algoritam sa sposobnošću nelinearne i ograničene te optimizacije bez potrebe za korištenjem derivacija funkcija. U ovom je radu predloženo i uspoređeno šest stohastičkih algoritama.

Prva tri predložena algoritma pripadaju skupini metaheurističkih iterativnih algoritama kod kojih buduće stanje, odabrani razmještaj osjetila, u svakom koraku ovisi samo o razmještaju u trenutnoj iteraciji algoritma. Navedeni algoritmi često su korišteni i uspoređivani za razne primjene [82, 83, 84].

Prvi od njih, Nelder-Mead metoda (NMM) (eng. Nelder-Mead Method) [85], pri optimizaciji koristi princip simpleksa (eng. simplex), jednostavni poligon s brojem vrhova za jedan većim od dimenzije optimizacijskog problema, odnosno broju nezavisnih varijabli koje se optimizira. Vrijednost optimizacijske funkcije računa se za svaki od vrhova simpleksa te se potom izračuna centroid skupa vrhova bez vrha s najgorom, u slučaju minimizacije najvećom, vrijednosti. Centroid se potom koristi u operacijama nad simpleksom. Operacije koje se redom izvode su refleksija (eng. reflection), širenje (eng. expansion), stezanje (eng. contraction) i smanjenje (eng. shrink) nad simpleksom [85].

Algoritam uspona uz brijeg (HCA) (eng. Hill-Climbing Algorithm) [86] drugi je odabrani algoritam. Ovaj će algoritam sigurno pronaći rješenje za konveksne probleme, ali za ostale može, umjesto globalnog, naći lokalni optimum. Iz tog razloga, njegova uspješnost kod nekonveksnih problema ovisi o nasumičnom odabiru početnih vrijednosti.

Treći odabrani algoritam, simulirano kaljenje (SA), rješava problem pronalaska globalnog optimuma za nekonveksne probleme koji se pojavio kod algoritma uspona uz brijeg (HCA). Simulirano kaljenje (SA) [87, 88] koristi metaheuristički pristup pri globalnoj optimizaciji u velikom prostoru pretrage, primjerice optimizacijskih funkcija s puno nezavisnih varijabli.

Uz navedena tri metaheuristička iterativna algoritma odabrana su i tri metaheuristička genetska optimizacijska algoritma. Sva tri algoritma temelje se na optimizaciji korištenjem populacije te na inteligenciji roja (eng. swarm intelligence) i idejama preuzetim iz prirode. Sva tri algoritma korištena su i uspoređena za planiranje putanje mobilnog robota [89], za rješavanje problema razmještaja osjetila [69], ali i za niz drugih problema [90, 91, 92].

Algoritam optimizacije roja čestica (PSO) [93] najčešće je korišteni optimizacijski algoritam temeljen na inteligenciji roja. Ovaj algoritam koristi čestice koje predstavljaju jato ptica ili riba koje pretražuju prostor koristeći kognitivnu i društvenu inteligenciju te brzinu svake od čestica.

Kao primjer novijeg optimizacijskog algoritma odabran je algoritam umjetnog roja pčela (ABC) [94]. Algoritam umjetnog roja pčela (ABC) preuzeo je ideje od pčela koje pretražuju prostor za grupom cvijeća gdje mogu sakupiti čim veću količinu biljnog nektra.

Treći algoritam, algoritam vatrometa (FWA) [95, 96], najrecentniji je od svih predloženih algoritama. U svakoj se iteraciji za svaku od čestica populacije radi lokalna pretraga oko njene pozicije.

Za svaki od algoritama odabrana su tri skupa parametara unutar vrijednosti navedenih u njihovim izvornim radovima ili radovima koji su te algoritme koristili. Detaljnija analiza utjecaja pojedinačnih parametara algoritama nije provedena. Za Nelder-Mead metodu (NMM) zadane su vrijednosti: $alpha = 0{,}1$ za koeficijent refleksije, $gamma = 0{,}3$ za koeficijent širenja, te $rho = sigma = 0{,}2$ za koeficijente stezanja i smanjenja. Za algoritme uspona uz brijeg (HCA) i simuliranog kaljenja (SA) zadana je vrijednost koraka lokalne pretrage $delta = 0{,}5$. Ispitivanjem različitih veličina rojeva na problemima različitih dimenzionalnosti odabrana je ista veličina roja $N = 10$ za sva tri genetska algoritma. Za algoritam optimizacije roja čestica (PSO) vrijednosti kognitivne i socijalne komponente su postavljeni na $C1 = C2 = 1{,}0$ dok je vrijednost inercije $w = 0{,}7$. Za algoritam vatrometa (FWA) koeficijenti pojačanja i umanjenja su postavljeni na $Ca = Cr = 10$.

Prva tri algoritma analiziraju samo jedno rješenje po iteraciji dok algoritmi temeljeni na populaciji analiziraju čitav raspon rješenja po iteraciji, ovisno o broju čestica. Algoritmi temeljeni na populaciji daju bolje rezultate od algoritama temeljenih samo na jednom rješenju, ali uz očekivano duže vrijeme obrade [97]. Algoritam uspona uz brijeg je algoritam lokalnog pretraživanja (eng. local search) te kao takav ima nedostatak mogućeg odabira lokalnog umjesto globalnog optimuma. Preostalih pet algoritama pripadaju kategoriji algoritama globalnog pretraživanja te kao takvi imaju mogućnost izlaska iz nepovoljnih lokalnih optimuma, ali uz veći računski trošak [98, 99].

Kako kod problema s nepoznatim rješenjima nije moguće provesti optimizaciju do zadane optimalne vrijednosti (eng. fixed-target), potrebno je ograničiti „cijenu" izvođenja algoritama koje se uspoređuje (eng. fixed-cost) [100]. Jedan od mogućih pristupa je

ograničavanje broja izvršavanja optimizacijske funkcije koje svaki od algoritama izvrši za zadani ispitni slučaj [101]. Kod ovog pristupa fokus se stavlja na ostvareni rezultat, pokrivenosti prostora, nakon provedene optimizacije. Na izvršavanje, odnosno izračun, optimizacijske funkcije opada većina vremena izvođenja, neovisno o optimizacijskom algoritmu koji se koristi. Razlike u vremenu izvođenja ovise o samom algoritmu i njegovoj implementaciji.

Kako se ispitni slučajevi razlikuju po dimenziji prostora te vrsti i broju osjetila koja se razmještaju, različit je i broj nezavisnih varijabli, pozicija i kuteva osjetila, korištenih u optimizacijskoj funkciji. Broj izvršavanja optimizacijske funkcije postavljen je na $E = 100 \cdot D$ gdje je D broj nezavisnih varijabli. Koeficijent od 100 izvršavanja po broju nezavisnih varijabli dobiven je empirijski u preliminarnom istraživanju. Ispitani su rezultati izvršavanja optimizacijskih algoritama na manjem skupu ispitnih slučajeva s korakom od 25 u broju izvršavanja po broju nezavisnih varijabli u domeni između 25 i 500 te korakom od 50 u domeni između 500 i 1000 izvršavanja optimizacijske funkcije. Iznad 100 izvršavanja optimizacijske funkcije po broju nezavisnih varijabli ne postoji statistički značajna razlika u ostvarenoj pokrivenosti prostora.

7.4. Korišteni računalni resursi

Ispitivanja funkcionalnosti modela provedena su na dva računala sličnih karakteristika. Oba su računala imala procesor „I5-4460" i $8\,GB$ radne memorije. Jedno od računala koristilo je Ubuntu 16.04 operacijski sustav dok je drugo koristilo Ubuntu-Mate 18.04. Na navedenim računalima pokrenut je manji broj ispitnih slučajeva za provjeru i potvrdu ispravnosti modela.

Svaki od šest odabranih optimizacijskih algoritama ispitan je nad svih 108 ispitnih slučajeva. Svaki od ispitnih slučajeva s odabranim algoritmom ponovljen je 100 puta kako bi se umanjio utjecaj urođene stohastičke prirode optimizacijskih algoritama. Zbog broja i računalne složenosti ispitnih slučajeva bilo je potrebno iskoristiti veću količinu računalnih, primarno procesorskih, resursa. Navedene resurse te obradu svih ispitnih slučajeva omogućilo je superračunalo „Bura" Sveučilišta u Rijeci [79].

Superračunalo „Bura" zasniva se na hibridnoj računalnoj arhitekturi koja se sastoji od višeračunalnih, heterogenih i višeprocesorskih sustava. U ovom istraživanju je korišten

višeračunalni sustav grozda računala sastavljen od 288 računalnih čvorova s dva procesora „Xeon E5" po čvoru. Svaki čvor ima ukupno 24 fizičke jezgre uz podršku za 48 virtualnih jezgri i 64 GB radne memorije. Superračunalo pogoni Red Hat Enterprise Linux 7 i Slurm Workload Manager.

U ispitivanju je iskorišteno 40 čvorova uz preduvjet samostalnog izvršavanja po jednog ispitnog slučaja na svakoj od 24 fizičke jezgre. Time je umanjen utjecaj operacijskog sustava na prebacivanje zadaća između dretvi te ostalih gubitaka u prebacivanju konteksta (eng. context switching). Ovime, ali i ponavljanjem ispitivanja 100 puta za svaki ispitni slučaj, osigurano je da rezultantna mjerenja vremena budu relevantna i upotrebljiva pri usporedbi algoritama.

8. Poglavlje

REZULTATI EKSPERIMENATA

8.1. Iscrpno pretraživanje

Kako je opisano u poglavlju 1., ne postoji općenito analitičko rješenje za razmještaj mjernih osjetila u prostoru [27, 29, 25, 35, 30]. Shodno tome, nije moguće analitički odrediti optimalan razmještaj osjetila za svaki ispitni slučaj i koristiti ga kao kriterij za određivanje točnosti (eng. ground-truth).

Optimalno rješenje za diskretne prostore pretrage je moguće odrediti korištenjem pristupa iscrpnog pretraživanja (ES) (eng. Exhaustive Search). Unutar diskretnog prostora pretrage određenog rasterizacijskom vrijednosti r_e iscrpnim se pretraživanjem (ES) ispita svaki mogući razmještaj osjetila i odabere se najbolji, onaj koji ostvaruje najveću pokrivenost prostora.

Model razmještaja osjetila predložen u ovoj disertaciji koristi kontinuirane vrijednosti za pozicije i orijentacije osjetila. Kako iscrpno pretraživanje (ES) pretražuje skup diskretnih pozicija i orijentacija, a vrijednosti koje se koriste su kontinuirane, pokrivenost dobivena iscrpnim pretraživanjem (ES) nije nužno optimalna.

Radi vremenske složenosti iscrpnog pretraživanja (ES) dodatno je razvijen i ispitan pristup rekurzivnog iscrpnog pretraživanja (RES) kod kojeg se prostor obilazio rekurzivno u potrazi za optimalnim rješenjem. Ovaj pristup kombinira ideje iscrpnog pretraživanja (ES) i pohlepnog pretraživanja (GS). Kao što je to slučaj s pohlepnim pretraživanjem (GS) postoji problem odabira lokalnog, umjesto globalnog, ekstrema.

Tijekom rekurzivnog iscrpnog pretraživanja (RES) vrijednosti koordinata potencijal-

nih pozicija razmaknute su za vrijednost rasterizacije prostora r_e. Inicijalna vrijednost rasterizacije $r_e = 1\,m$. Za svaku od potencijalnih pozicija izračunata je vrijednost pokrivenosti C te je odabrana pozicija, ili pozicije, najveće vrijednosti pokrivenosti. Prostor pretrage se potom smanji te njegova veličina iznosi $\frac{r_e}{2}$ po svakoj od osi koordinatnog sustava prostora oko odabrane pozicije. Po smanjenju prostora pretrage smanjuje se i vrijednost r_e za jedan red veličine, odnosno njena se vrijednost dijeli s deset. Izvođenje se prekida kada širina prostora pretrage padne ispod vrijednosti preciznosti brojeva s pomičnim zarezom (eng. floating-point) $r_e <= 1 \cdot 10^{-8}$.

Tablica 8.1: Rezultati pokrivenosti prostora C i broja izvršavanja optimizacijske funkcije E iscrpnog pretraživanja (ES) i rekurzivnog iscrpnog pretraživanja (RES) za ispitni slučaj s jednim svesmjernim osjetilom razmještanim u dvodimenzionalne i trodimenzionalne modele sva tri ispitna prostora s rasterizacijskom vrijednosti $r_v = 0{,}1\,m$

		Ispitni prostor 1		Ispitni prostor 2		Ispitni prostor 3	
		2D	3D	2D	3D	2D	3D
ES	C	0.1446	-	0.1191	-	0.1359	-
	E	221301	$1.107 \cdot 10^9$	210197	$1.133 \cdot 10^9$	205530	$1.028 \cdot 10^9$
RES	C	0.1446	0.0795	0.1191	0.0814	0.1359	0.0742
	E	1567	21230	1533	17051	1534	16874

Ova dva pristupa su ispitana i uspoređena na ispitnim slučajevima razmještaja jednog svesmjernog osjetila u dvodimenzionalnim i trodimenzionalnim modelima sva tri prostora s rasterizacijskom vrijednosti $r_v = 0{,}1\,m$. Iscrpno pretraživanje (ES) provedeno je s korakom $r_e = 0{,}01\,m$ dok je za rekurzivno iscrpno pretraživanje (RES) inicijalna vrijednost $r_e = 1\,m$. U tablici 8.1 dan je broj voksela svakog od modela prostora te vrijednosti ostvarenih pokrivenosti prostora C i broja izvršavanja optimizacijske funkcije E za oba pristupa.

Kako se može očekivati isto prosječno vrijeme izvođenja jednog izvršavanja optimizacijske funkcije, umjesto vremena izvršavanja algoritmi su uspoređeni po broju izvršavanja optimizacijske funkcije. Pristupi iscrpnog pretraživanja (ES) i rekurzivnog iscrpnog pretraživanja (RES) su paralelizirani te ih nije moguće međusobno usporediti temeljem vremena izvođenja.

Svaki od ispitnih slučajeva ispitan s rekurzivnim ispitnim pretraživanjem (RES) pokrenut je na 480 fizičkih jezgri procesora, 20 čvorova, superračunala „Bura”. Vrijeme izvođenja za trodimenzionalne modele ispitnih slučajeva kretalo se između 6.5 i 14.5 sati.

Procijenjeno vrijeme izvođenja za iscrpno pretraživanje (ES) na istim ispitnim slučajevima i s istim računalnim resursima kreće se između četiri i 11 godina.

Iz dobivenih podataka vidljivo je da između dva navedena pristupa kod dvodimenzionalnih modela prostora nema razlike u ostvarenoj pokrivenosti. Kod trodimenzionalnih modela prostora, a zbog ranije navedenog problema s vremenom izvođenja, rezultati za ostvarenu pokrivenost prostora za pristup iscrpnog pretraživanja (ES) nisu prikazani. Za ovaj set odabranih ispitnih slučajeva dobivene vrijednosti pokrivenosti prostora smatraju se optimalnima, odnosno referentnima za daljnju usporedbu. Razmještaj iscrpnim pretraživanjem (ES) i rekurzivnim iscrpnim pretraživanjem (RES) proveden je samo s jednim svesmjernim osjetilom. Ispitni slučajevi s većim brojem osjetila ili s usmjerenim osjetilima računalno su značajno složeniji te ih ne bi bilo moguće provesti u stvarnom vremenu.

8.2. Usporedba iscrpnog pretraživanja i stohastičkih algoritama

Pristup razmještaju osjetila za pokrivanje zatvorenoga prostora uz korištenje stohastičkih optimizacijskih algoritama uspoređen je s pristupom rekurzivnog iscrpnog pretraživanja (RES) za ranije navedene ispitne slučajeve, tablica 8.1.

Tablica 8.2: Rezultati pokrivenosti prostora C rekurzivnog iscrpnog pretraživanja (RES) i šest odabranih algoritama za ispitni slučaj s jednim svesmjernim osjetilom razmještanim u dvodimenzionalne i trodimenzionalne modele sva tri ispitna prostora s rasterizacijskom vrijednosti $r_v = 0{,}1\,m$

	Ispitni prostor 1		Ispitni prostor 2		Ispitni prostor 3	
	2D	3D	2D	3D	2D	3D
RES	0.1446	0.0795	0.1191	0.0814	0.1359	0.0742
ABC	0.1440	0.0792	0.1188	0.0812	0.1350	0.0737
FWA	0.1430	0.0765	0.1164	0.0794	0.1346	0.0723
PSO	**0.1445**	**0.0795**	**0.1188**	**0.0814**	**0.1357**	**0.0740**
HCA	0.1442	0.0790	0.1183	0.0809	0.1353	0.0735
NMM	0.1271	0.0671	0.0968	0.0709	0.1150	0.0635
SA	0.1351	0.0694	0.1082	0.0713	0.1244	0.0670

Rezultati pokrivenosti prostora C za svih šest odabranih optimizacijskih algoritama za iste ispitne slučajeve danu su u tablici 8.2. Algoritam optimizacije roja čestica (PSO)

ostvario je najbolje rezultate pokrivenosti prostora za ove ispitne slučajeve. Relativna pogreška u pokrivenosti prostora za odabrani razmještaj između rekurzivnog iscrpnog pretraživanja (RES) i algoritma optimizacije roja čestica (PSO) je u prosjeku 0,14%. Kako je objašnjeno u prošlom poglavlju, svi algoritmi limitirani su brojem izvršavanja optimizacijske funkcije. U ovim ispitnim slučajevima izvršeno je 200 izvršavanja optimizacijske funkcije za dvodimenzionalne modele prostora te 300 izvršavanja za trodimenzionalne prostore. Uz pretpostavku istog prosječnog vremena izvođenja optimizacijske funkcije korištenjem optimizacijskih algoritama dobiva se ubrzanje od oko 7,5 puta za dvodimenzionalne modele te između 55 i 70 puta za trodimenzionalne modele prostora.

8.3. Usporedba optimizacijskih algoritama

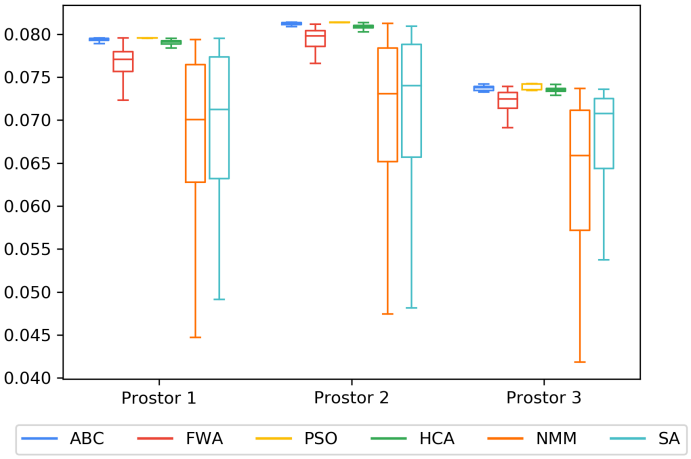
Za 100 ponavljanja svakog ispitnog slučaja izračunate su najmanje, srednje te najveće vrijednosti pokrivenosti prostora i vremena izvršavanja. U tablicama B.1, B.2 i B.3 u prilogu B. dane su srednje vrijednosti pokrivenosti dobivene optimizacijom koristeći svih šest algoritama za sva tri ispitna prostora uz variranje dimenzija modela prostora, vrijednosti rasterizacijske vrijednosti te brojem i vrstom osjetila. Vremena izvršavanja optimizacijskih algoritama, kao drugi bitan čimbenik, dana su u prilogu B. u tablicama B.4, B.5 i B.6. Vrijednosti su navedene u sekundama i zaokružene na najbliži cijeli broj.

Kako je glavni fokus ovog istraživanja na razmještaju osjetila u trodimenzionalnom modelu prostora, na sljedećim slikama dan je prikaz kutijastih dijagrama (eng. boxplot) pokrivenosti prostora za 18 ispitnih slučajeva koji sadrže trodimenzionalne modele prostora s najmanjom rasterizacijskom vrijednosti od $0{,}1\,m$.

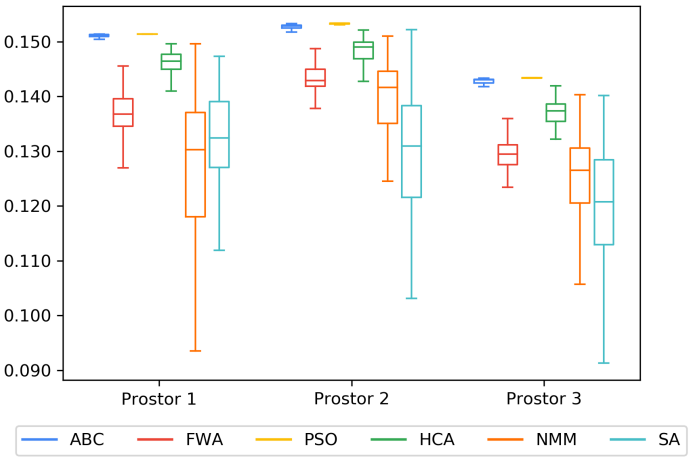
Na svakoj od slika prikazani su rezultati ispitivanja za svih šest algoritama za svaki od triju ispitnih prostora. Na slikama 8.1 i 8.2 prikazani su kutijasti dijagrami rezultata pokrivenosti prostora, a na slikama 8.3 i 8.4 kutijasti dijagrami vremena izvršavanja.

Na slici 8.1 a) dan je prikaz za jedno, na slici 8.1 b) za sva te na slici 8.1 c) za tri usmjerene osjetila Estimote BLE. Na slici 8.2 prikazani ostvareni rezultati pokrivenosti usmjerenog osjetilo Stereolabs ZED. Ostvareni rezultati pokrivenosti jednog osjetila prikazani su na slici 8.2 a), na slici 8.2 b) za dva osjetila te na slici 8.2 c) za tri usmjerena osjetila. Isti opis broja i vrsti osjetila vrijedi i za slike 8.3 i 8.4.

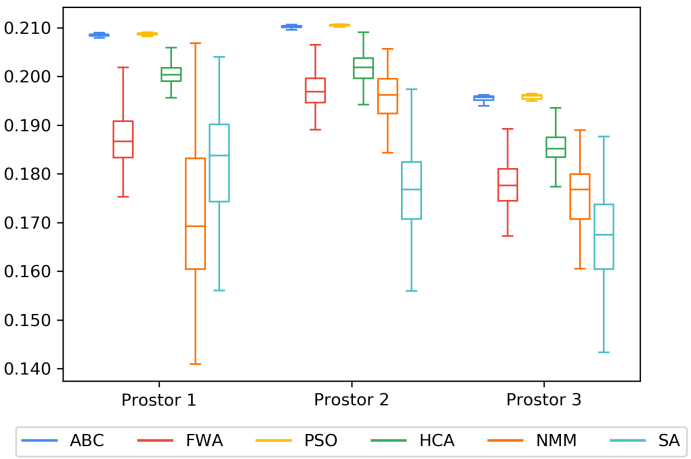
U svim prikazanim primjerima algoritam umjetnog roja pčela (ABC) ostvaruje naj-



(a) Jedno osjetilo

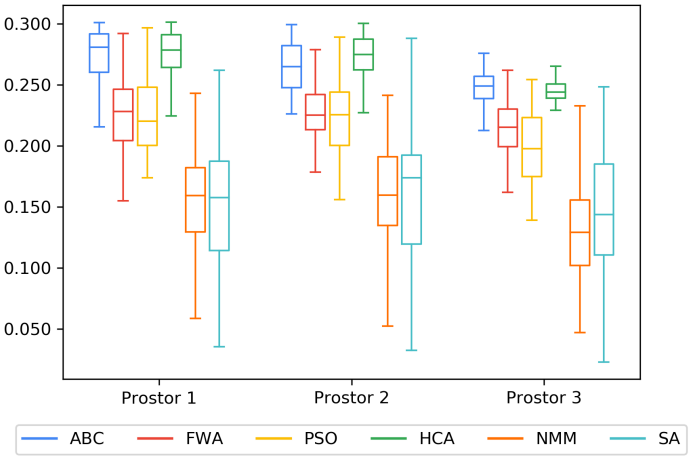


(b) Dva osjetila

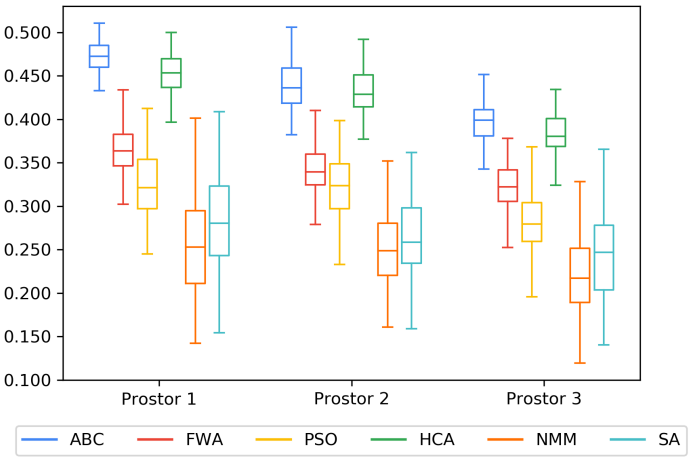


(c) Tri osjetila

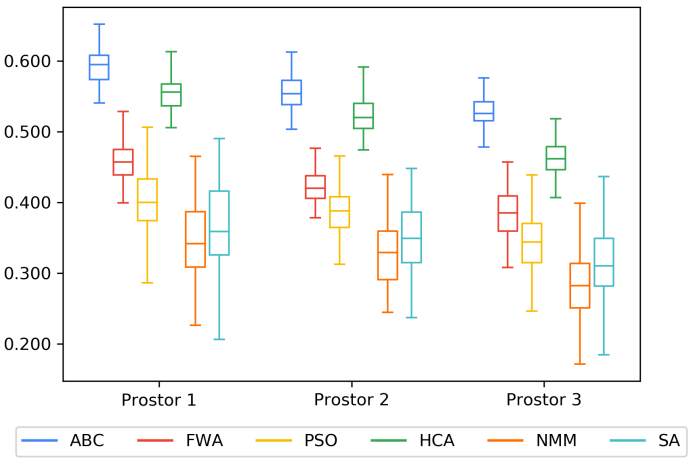
Slika 8.1: Kutijasti dijagrami rezultata pokrivenosti prostora za podskup ispitnih slučajeva s trodimenzionalnim modelima prostora za razmještaj jednog, dva i tri svesmjerna osjetila



(a) Jedno osjetilo

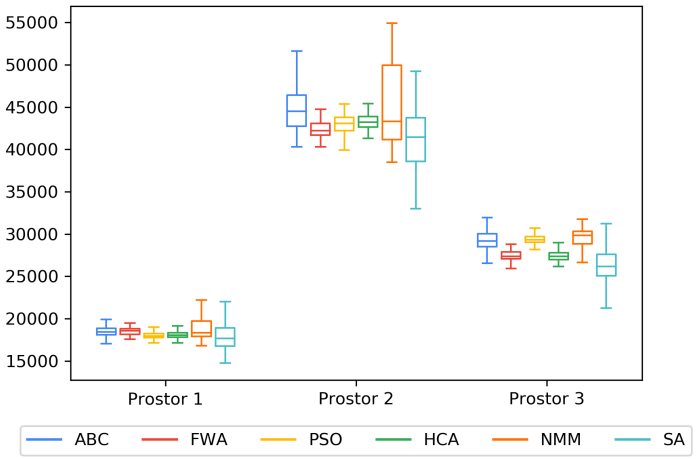


(b) Dva osjetila

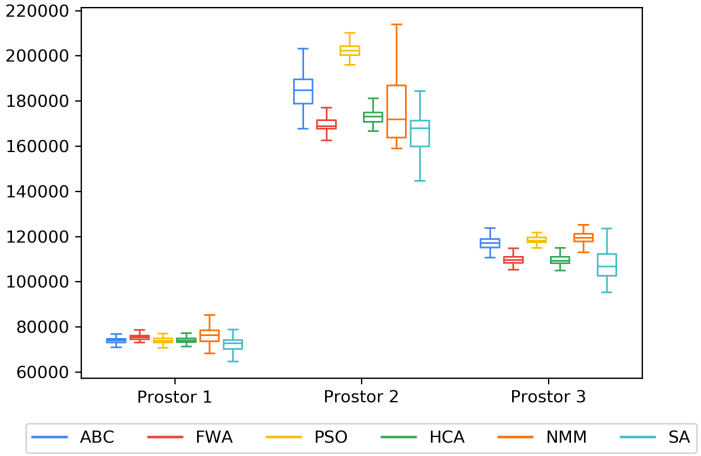


(c) Tri osjetila

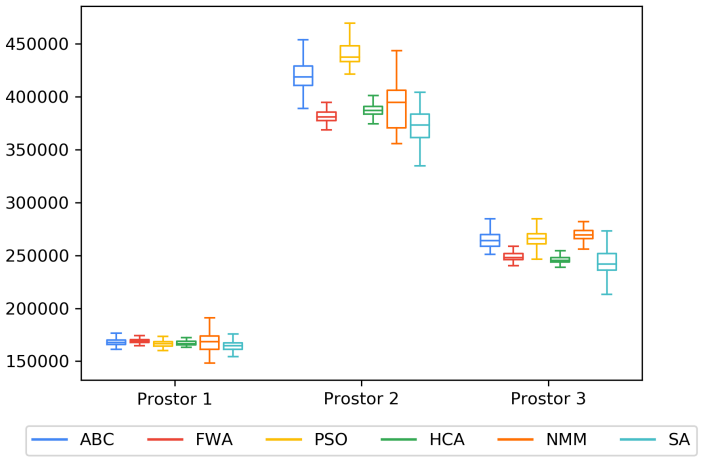
Slika 8.2: Kutijasti dijagrami rezultata pokrivenosti prostora za podskup ispitnih slučajeva s trodimenzionalnim modelima prostora za razmještaj jednog, dva i tri usmjerena osjetila



(a) Jedno osjetilo

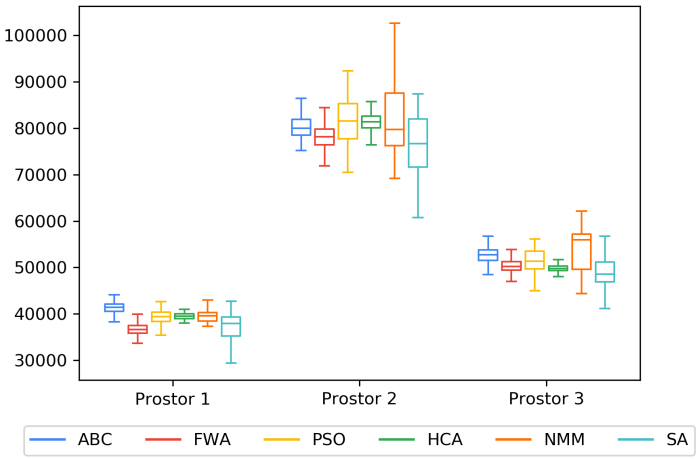


(b) Dva osjetila

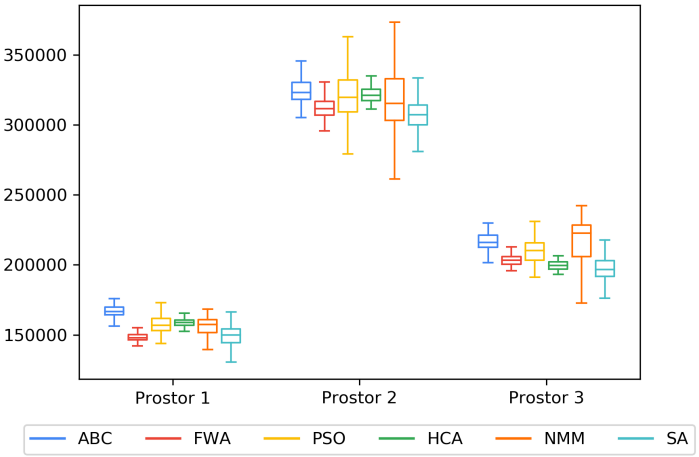


(c) Tri osjetila

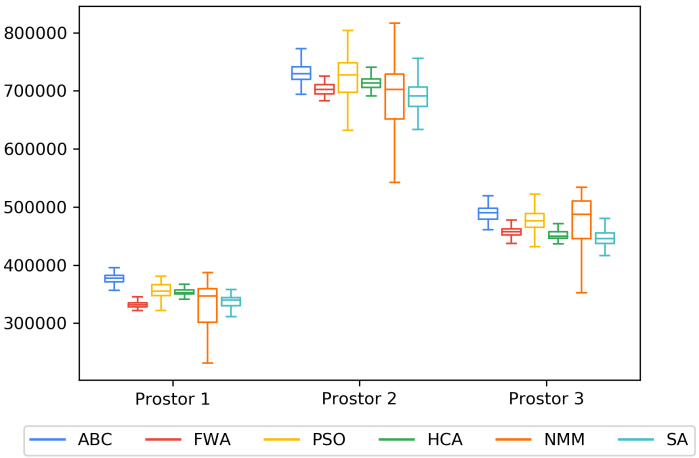
Slika 8.3: Kutijasti dijagrami vremena izvršavanja za podskup ispitnih slučajeva s trodimenzionalnim modelima prostora za razmještaj jednog, dva i tri svesmjerna osjetila



(a) Jedno osjetilo



(b) Dva osjetila



(c) Tri osjetila

Slika 8.4: Kutijasti dijagrami vremena izvršavanja za podskup ispitnih slučajeva s trodimenzionalnim modelima prostora za razmještaj jednog, dva i tri usmjerena osjetila

bolje rezultate pokrivenosti prostora. Iza njega slijedi algoritam optimizacije roja čestica (PSO) te algoritam uspona uz brijeg (HCA). Algoritam optimizacije roja čestica (PSO) za svesmjerna osjetila ostvaruje rezultate koji su slični algoritmu umjetnog roja pčela (ABC), dok za usmjerena osjetila ostvaruje znatno niže rezultate. Moguć razlog je u broju nezavisnih varijabli koje se koriste pri optimizaciji jer za svako usmjereno osjetilo postoje dodatne dvije nezavisne varijable, kutevi osjetila. U daljnjim će istraživanjima utjecaj broja nezavisnih varijabli te uzrok ove nastale razlike biti detaljno istražen. Od preostalih algoritama, algoritam vatrometa (FWA) daje najbolje rezultate pokrivenosti prostora, a nakon njega slijede simulirano kaljenje (SA) i Nelder-Mead metoda (NMM).

Iz kutijastih dijagrama vremena izvršavanja nije moguće donijeti zaključke o razlikama u brzini izvođenja algoritama. Za obje vrste osjetila simulirano kaljenje (SA) uglavnom postiže najmanje medijalno vrijeme izvođenja, ali s većom varijancom u odnosu na dio drugih algoritama. Medijalne vrijednosti vremena izvođenja nešto veće od simuliranog kaljenja (SA) ostvaruju algoritam vatrometa (FWA), koji je u jednom slučaju i bolji od simuliranog kaljenja (SA), te Nelder-Mead metoda (NMM).

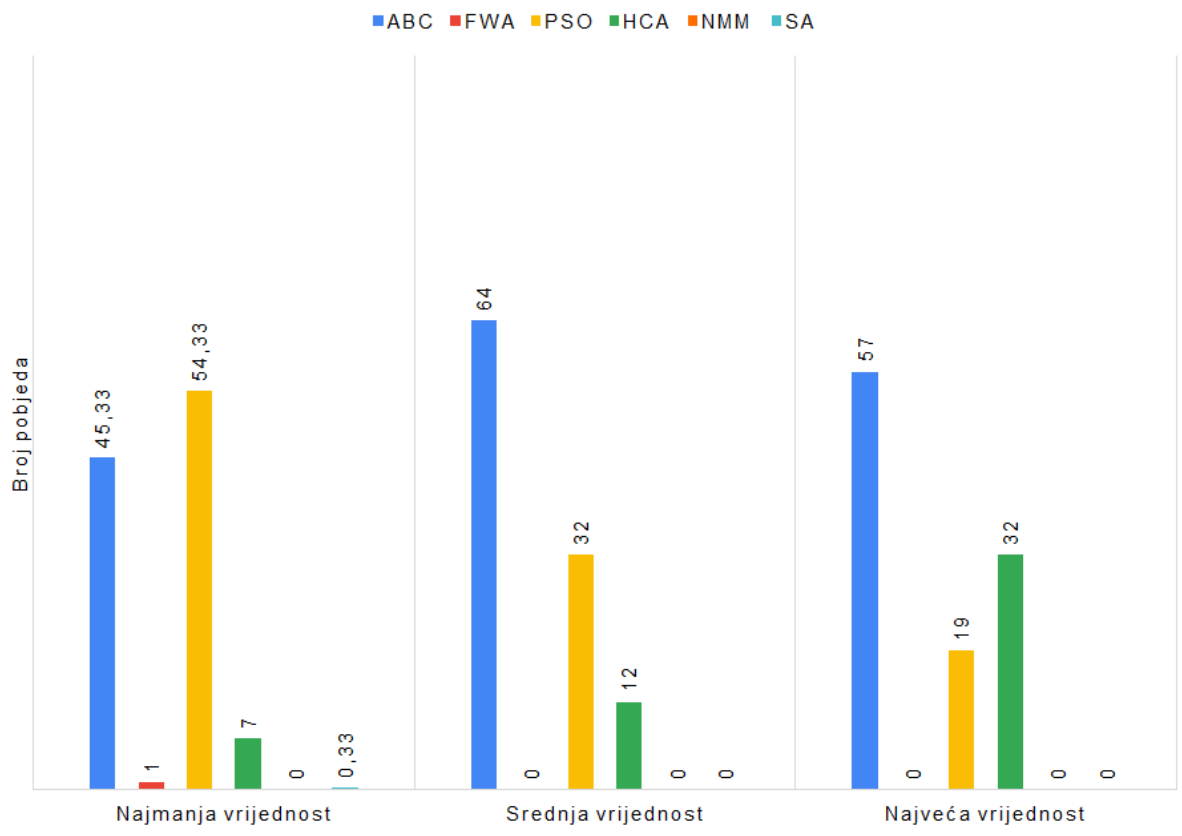
Kako optimalan razmještaj za svaki ispitni slučaj nije poznat, u nastavku će se algoritmi usporediti na tri načina: brojanjem pobjeda [102, 69], brojanjem statistički značajnih pobjeda [102] te statističkom analizom rezultata svih ispitnih slučajeva.

8.3.1. Brojanje pobjeda

Prvi pristup usporedbi algoritama je brojanje pobjeda. Za svaki ispitni slučaj algoritmu koji postigne najveću pokrivenost prostora, brojač pobjeda se poveća za jedan. Ako dva ili više algoritma postignu isti rezultat, svakom se od algoritama pribroji jednak udio te pobjede. Pobjede se broje u tri osnovne kategorije, najmanja, srednja i najveća vrijednost 100 ponavljanja [103].

Za pristup brojanja pobjeda pokrivenosti prostora u kategoriji najmanje vrijednosti, slika 8.5, algoritam optimizacije roja čestica (PSO) ostvario je najviše pobjeda, 54,33, odnosno za najviše ispitnih slučajeva je ostvario najveći rezultat. Iza njega slijedi algoritam umjetnog roja pčela (ABC) s 45,33 pobjeda te nakon njega ostala četiri algoritma sa značajno nižim rezultatima.

Za kategoriju srednje vrijednosti, algoritam umjetnog roja pčela (ABC) sa 64 pobjede



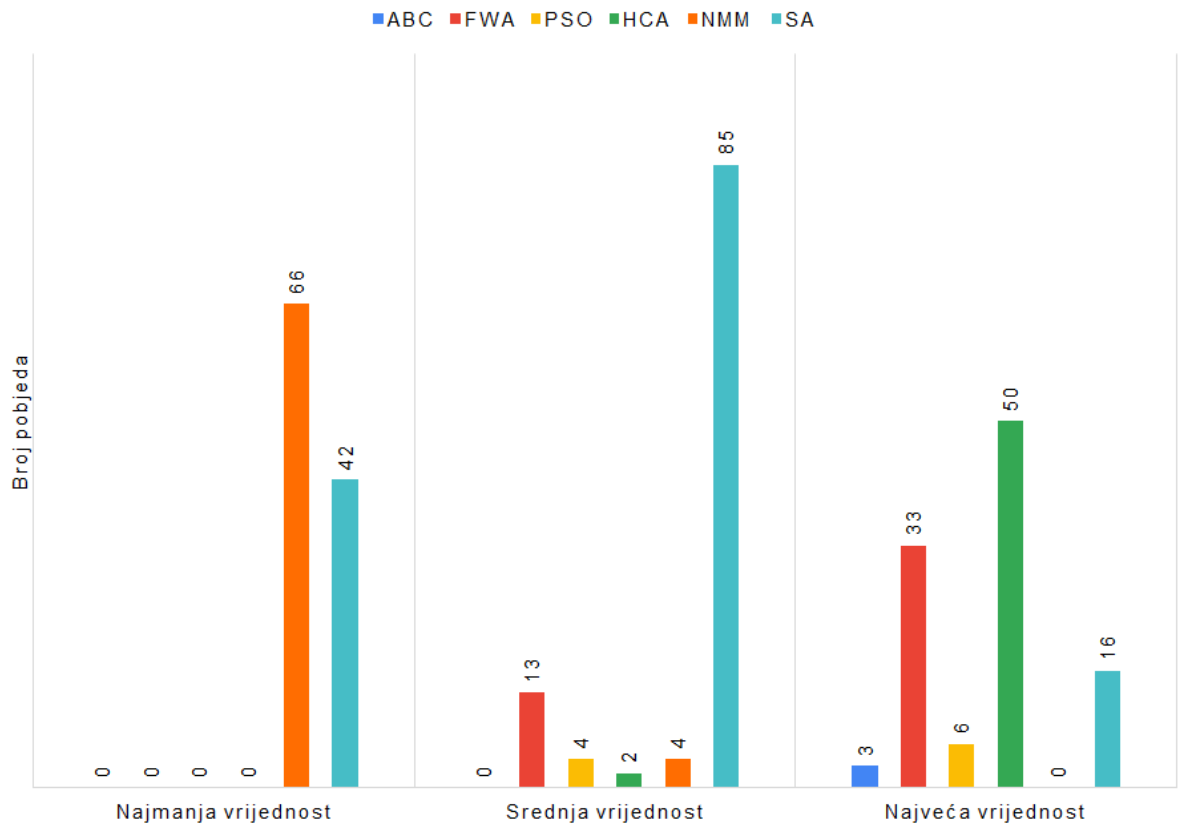
Slika 8.5: Rezultati brojanja pobjeda pokrivenosti prostora

se pokazuje kao najbolji, a slijede ga algoritam optimizacije roja čestica (PSO) s 32 pobjede i algoritam uspona uz brijeg (HCA) s 12 pobjeda. Preostala tri algoritma nisu se pokazali najboljim u kategoriji srednje vrijednosti niti za jedan od ispitnih slučajeva.

U trećoj kategoriji, kategoriji najveće vrijednosti, ponovno je algoritam umjetnog roja pčela (ABC) ostvario najviše pobjeda, 57, dok su algoritam optimizacije roja čestica (PSO) s 19 pobjeda i algoritam uspona uz brijeg s 32 pobjede zamijenili mjesta u poretku. Ostatak algoritama ponovno nije ostvario niti jednu od pobjeda.

Za vrijednost pokrivenosti prostora, algoritam umjetnog roja pčela (ABC) pokazao se najboljim, osim u kategoriji najmanje vrijednosti u kojoj je algoritam optimizacije roja čestica (PSO) dao bolje rezultate. Vidljivo je da, iako algoritam optimizacije roja čestica (PSO) daje dobre rezultate pri analizi rezultata najmanje pokrivenosti, rijetko dostiže najveće vrijednosti. Također, algoritam uspona uz brijeg (HCA) daje dobre rezultate za kategoriju najveće pokrivenosti što je pokazatelj dobrog odabira vrijednosti.

Kod usporedbe vremena izvođenja, pobjedom se smatra najmanja vrijednost u svakoj



Slika 8.6: Rezultati brojanja pobjeda vremena izvršavanja

kategoriji. Za kategoriju najmanje vrijednosti, slika 8.6, samo su Nelder-Mead metoda (NMM) i simulirano kaljenje (SA) ostvarili pobjede. Od simuliranog kaljenja (SA) s 42 pobjede, malo se bržom pokazala Nelder-Mead metoda (NMM) sa 66 pobjeda.

Za kategoriju srednje vrijednosti vremena izvođenja, simulirano se kaljenje (SA) pokazalo značajnije bržim od ostalih metoda s 85 pobjeda te ga slijedi algoritam vatrometa (FWA) s 13 pobjeda. Algoritam umjetnog roja pčela (ABC) nije ostvario niti jednu pobjedu u ovoj kategoriji.

U trećoj se kategoriji značajno mijenjaju stvari te algoritam uspona uz brijeg (HCA) ostvaruje najviše pobjeda, 50, a slijedi ga algoritam vatrometa (FWA) s 33 te simulirano kaljenje (SA) sa 16 pobjeda. Iako je Nelder-Mead metoda (NMM) ostvarila najveći broj pobjeda u kategoriji najmanje vrijednosti, za ovu kategoriju ostvaruje nula pobjeda.

Iz navedenih rezultata po trima kategorijama može se zaključiti da je simulirano kaljenje (SA) u prosjeku najbrži algoritam, dok su Nelder-Mead metoda (NMM) i algoritam uspona uz brijeg (HCA) ostvarili najviše pobjeda u kategorijama najmanje i najveće

vrijednosti. Ovakvi rezultati za vrijeme izvođenja su očekivani i ponajviše ovise o vrsti algoritma. U skupu genetskih algoritama, algoritam vatrometa (FWA) pokazao se značajno bržim od preostala dva algoritma.

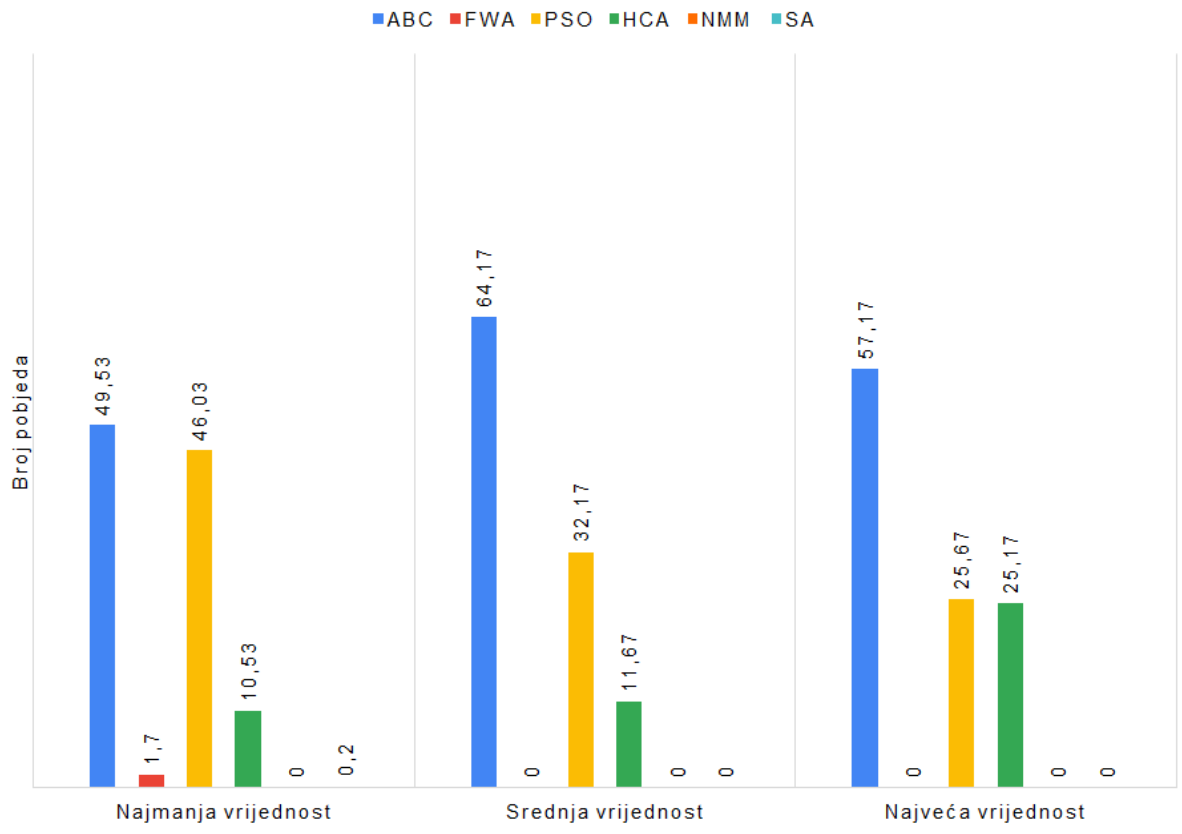
Iz ovih rezultata može se zaključiti da je algoritam umjetnog roja pčela (ABC) najbolji izbor za problem razmještaja osjetila ako vrijeme izvođenja ne predstavlja bitnu stavku pri odabiru algoritama. Ako je vrijeme izvođenja bitno, algoritam uspona uz brijeg (HCA) može biti primjeren zbog kompromisa ostvarene pokrivenosti prostora i vremena izvođenja. U konačnici, odabir algoritma ovisi o potrebama korisnika te ne postoji algoritam koji je najbolji u svim kategorijama.

8.3.2. Brojanje statistički značajnih pobjeda

U prethodnom pristupu brojanju pobjeda gleda se samo algoritam koji je ostvario najbolji rezultat. Time se zanemaruju algoritmi koji mogu imati različitu vrijednost, ali razlika nije statistički značajna. Za svaki se ispitni slučaj radi provjera statističke značajnosti razlika u vrijednosti pokrivenosti prostora ili vremena izvršavanja za svih 100 ispitivanja. Vrijednosti pokrivenosti prostora i vremena izvršavanja slijede normalnu razdiobu. Koristeći Mauchlyev test sferičnosti (eng. Mauchly's sphericity test) opovrgnuta je pretpostavka o sferičnosti vrijednosti.

Statistička analiza svakog ispitnog slučaja provedena je koristeći ANOVA (eng. Analysis of Variance) ponovljenih mjerenja (eng. Repeated measures ANOVA). Kako je pretpostavka o sferičnosti vrijednosti opovrgnuta, iskorištena je Greenhouse-Geisser korekcija. Ako razlika između rezultata nije statistički značajna svim se algoritmima pridoda jednak udio vrijednosti. Ako postoji statistički značajna razlika, primjenjuje se naknadna analiza (eng. post-hoc) koristeći Bonferroni korekciju. U prethodnom se pristupu pobjeda dijeli samo ako algoritmi postignu potpuno isti rezultat. Za razliku od toga, u pristupu brojanja pobjeda uz provjeru statističke značajnosti uzima se algoritam s najboljim rezultatom te svi algoritmi koji nisu statistički značajno različiti od njega. Tim se algoritmima, kao i u prethodnom pristupu, dodaje jednak udio te pobjede.

Ovaj pristup brojanju pobjeda pokrivenosti prostora, slika 8.7, nije dao rezultate koji su značajno različiti u odnosu na pristup brojanja pobjeda. U kategoriji najmanje vrijednosti algoritam umjetnog roja pčela (ABC) ostvario je, s 49,53 pobjede, viši rezultat



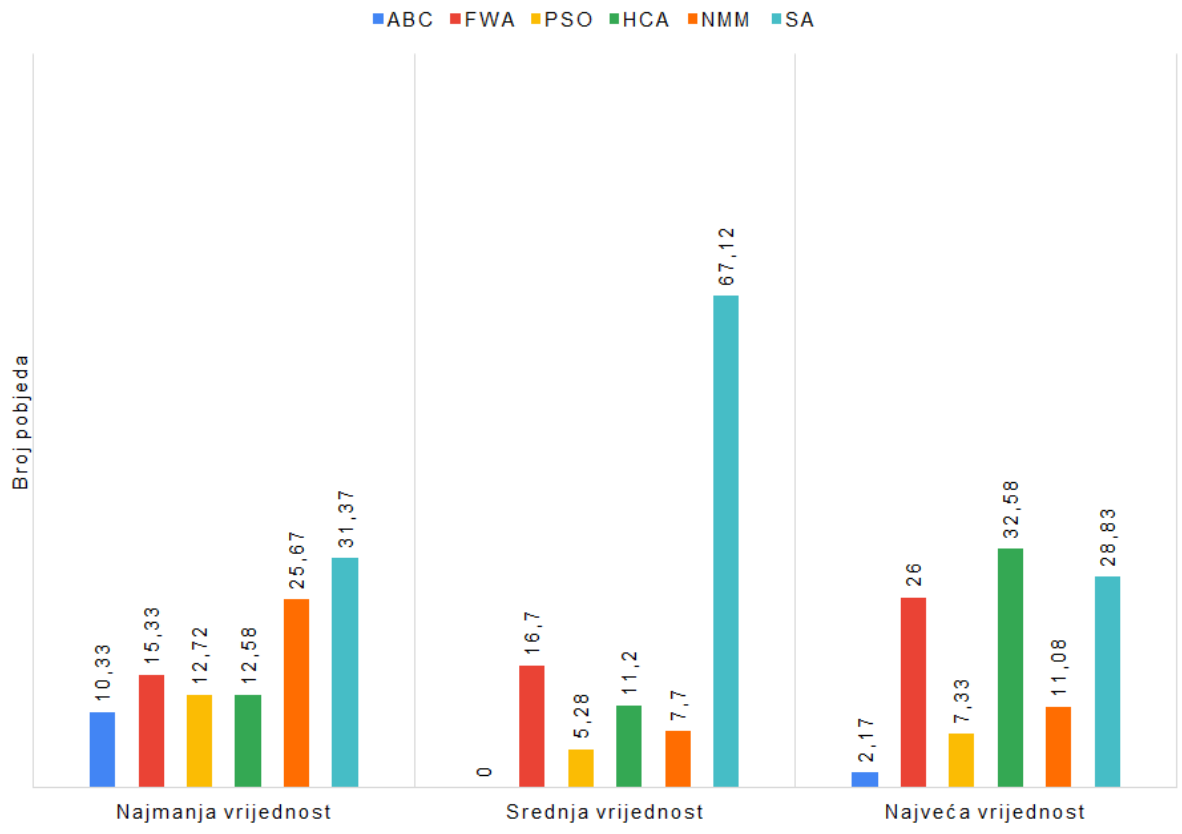
Slika 8.7: Rezultati brojanja statistički značajnih pobjeda pokrivenosti prostora

od algoritma optimizacije roja čestica (PSO). Algoritam optimizacije roja čestica (PSO) ostvario je manje, 46,03 pobjeda nego u prethodnom pristupu.

Rezultati u kategoriji srednje vrijednosti nemaju značajnih promjena dok je u kategoriji najveće vrijednosti promjena u poretku algoritma optimizacije roja čestica (PSO) s 25,67 pobjeda i algoritma uspona uz brijeg (HCA) s 25,17 pobjeda.

Za ovaj se pristup rezultati pokrivenosti prostora ne razlikuju se u odnosu na pristup brojanja pobjeda. Algoritam umjetnog roja pčela (ABC) se pokazao najboljim u svim kategorijama, dok je algoritam optimizacije roja čestica (PSO) ostvario drugi rezultat u svakoj od kategorija.

Kod vrijednosti vremena izvršavanja, slika 8.8, rezultati su značajno različiti u odnosu na pristup brojanja pobjeda. U kategoriji najmanje vrijednosti simulirano kaljenje (SA) ostvario je nešto niži rezultat nego kod pristupa brojanja pobjeda, ali je sada preuzeo prvo mjesto s 31,37 pobjeda. Algoritam uspona uz brijeg (HCA) sa 66 pobjeda pao na svega 25,67 te se ta razlika raspodijelila na ostala četiri algoritma koji su s 0 pobjeda narasli na



Slika 8.8: Rezultati brojanje statistički značajnih pobjeda vremena izvršavanja

više od 10.

U kategoriji srednje vrijednosti razlika nije značajna te je simulirano kaljenje (SA) i dalje najbolji sa 67,12 pobjeda, a broj pobjeda ostalih algoritama se nije značajno promijenio. Slična je situacija i za kategoriju najveće vrijednosti kod koje se smanjila razlika između algoritma uspona uz brijeg (HCA) te simuliranog kaljenja (SA) i algoritma vatrometa (FWA).

Iako se simulirano kaljenje (SA) pokazalo najboljim, a posebice u kategoriji srednje vrijednosti, ostali su algoritmi ostvarili bolje rezultate. U kategorijama najmanje i najveće vrijednosti razlike između algoritama su neznatne.

8.3.3. Statistička analiza srednjih vrijednosti rezultata

Posljednji pristup usporedbi je statistička analiza srednjih vrijednosti rezultata 100 ponavljanja za svih 108 ispitnih slučajeva. Vrijednosti deskriptivne statistike za svaki od algoritama dane su u tablici 8.3. U ovom pristupu ponovljena je analiza sferičnosti

podataka koristeći Mauchlyev test sferičnosti. Kako je Mauchlyev test sferičnosti opovrgnuo pretpostavku o sferičnosti podataka, korištena je ANOVA ponovljenih mjerenja s Greenhouse-Geisser korekcijom. Za pokrivenost prostora ANOVA ponovljenih mjerenja je potvrdila da postoji statistički značajna, $F(1{,}191,\ 127{,}446) = 125{,}859$, $p < 0{,}001$, razlika između rezultata algoritama. Naknadna analiza koristeći Bonferroni korekciju pokazala je statistički značajne razlike između svih parova algoritama osim između algoritma vatrometa (FWA) i algoritma optimizacije roja čestica (PSO) $p > 0{,}061$. Najboljim se pokazao algoritam umjetnog roja pčela (ABC) iza kojeg slijedi algoritam uspona uz brijeg (HCA). Nakon njih slijede algoritam optimizacije roja čestica (PSO) i algoritam vatrometa (FWA) iza kojih slijede simulirano kaljenje (SA) i Nelder-Mead metoda (NMM).

Tablica 8.3: Deskriptivna statistika algoritama za pokrivenost prostora

	ABC	HCA	FWA	PSO	SA	NMM
srednja vrijednost	0,3176	0,3036	0,2705	0,2668	0,2316	0,2167
standardna devijacija	0,1959	0,1819	0,1552	0,1454	0,1295	0,1183
najmanja vrijednost	0,0528	0,0523	0,0507	0,0533	0,0476	0,0439
25%	0,1505	0,1457	0,1392	0,1506	0,1289	0,1256
50%	0,2683	0,2688	0,2306	0,2420	0,2156	0,2020
75%	0,4415	0,4145	0,3628	0,3426	0,3122	0,2846
najveća vrijednost	0,8147	0,7896	0,6970	0,6776	0,5955	0,5596

Isto ispitivanje je ponovljeno i na podskupu ispitnih slučajeva koji sadrže samo trodimenzionalne modele prostora. Vrijednosti deskriptivne statistike na ovom podskupu dane su u tablici 8.4. ANOVA ponovljenih mjerenja s Greenhouse-Geisser korekcijom pokazala je da postoji i dalje statistički značajna, $F(1{,}092,\ 57{,}892) = 60{,}084$, $p < 0{,}001$, razlika između rezultata. Za razliku od rezultata naknadne analize nad cijelim skupom podataka, na podskupu ispitnih slučajeva s trodimenzionalnim modelima prostora utvrđeno je da ne postoji statistički značajna, $p = 0{,}018$, razlika između algoritma optimizacije roja čestica (PSO) i algoritma vatrometa (FWA).

ANOVA ponovljenih mjerenja s Greenhouse-Geisser korekcijom pokazala je da ne postoji statistički značajna, $F(1{,}795,\ 192{,}115) = 6{,}758$, $p > 0{,}999$, razlika između vremena izvođenja algoritama. Vidljiv je trend promjena razlika u vremenima algoritama kroz tri različita pristupa. Iz ovog je moguće zaključiti da, iako je simulirano kaljenje (SA) ostvarilo najviše pobjeda u kategoriji srednje vrijednosti, statistički značajna razlika između

Tablica 8.4: Deskriptivna statistika algoritama za podskup s trodimenzionalnim modelima prostora

	ABC	HCA	FWA	PSO	SA	NMM
srednja vrijednost	0,2530	0,2401	0,2071	0,2000	0,1713	0,1640
standardna devijacija	0,1596	0,1454	0,1151	0,0981	0,0894	0,0824
najmanja vrijednost	0,0528	0,0523	0,0507	0,0533	0,0476	0,0439
25%	0,1295	0,1256	0,1177	0,1293	0,1145	0,1101
50%	0,2081	0,2010	0,1772	0,1830	0,1464	0,1424
75%	0,3911	0,3747	0,3090	0,2773	0,2391	0,2217
najveća vrijednost	0,5917	0,5538	0,4580	0,4024	0,3670	0,3481

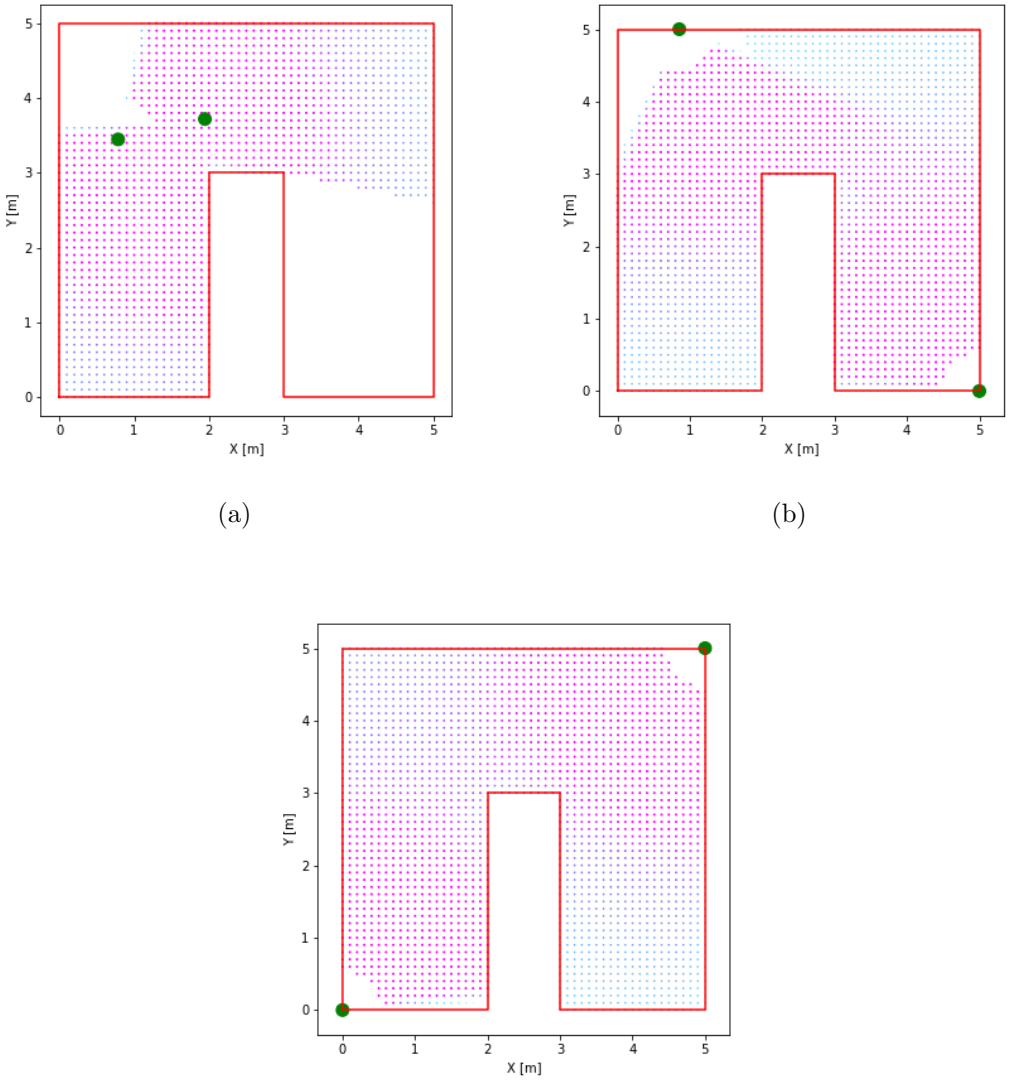
vremena izvođenja algoritama ne postoji. Kako veći dio vremena proračuna odlazi na izvršavanje optimizacijske funkcije, a ograničen je broj izvršavanja optimizacijske funkcije ovisno o ispitnom slučaju, kako je navedeno u ranijem poglavlju razlike između algoritama u vremenu izvođenja su neznatne i ovise samo o algoritmu i njegovoj implementaciji.

Ako se, kao i kod vrijednosti pokrivenosti prostora, ponovi ispitivanje na podskupu ispitnih slučajeva koji sadrže samo trodimenzionalne modele prostora, razlika između vremena izvođenja algoritama i dalje nije statistički značajna $F(1{,}855,\ 98{,}302) = 7{,}031$, $p > 0{,}999$.

8.4. Primjeri razmještaja osjetila

Kako se pokazao najboljim u prethodnim analizama temeljem pokrivenosti prostora, naredni primjeri prikazuju rezultate optimizacije algoritmom umjetnog roja pčela (ABC). U prvom primjeru, slika 8.9, prikazan je razmještaj dva usmjerena osjetila u dvodimenzionalnom modelu Ispitnog prostora 1. Od 100 ponavljanja odabrana su tri ponavljanja različitih vrijednosti pokrivenosti prostora.

Razmještaj na slici 8.9 a) je primjer razmještaja osjetila s manjom vrijednosti pokrivenosti prostora $C = 0{,}5557$. Na slici 8.9 b) dan je primjer razmještaja s većom vrijednosti pokrivenosti prostora $C = 0{,}6819$. U trećem primjeru, slika 8.9 c), dan je primjer razmještaja osjetila s najvećom vrijednosti pokrivenosti za ovaj ispitni slučaj $C = 0{,}7466$. U ovom je slučaju, za razliku od prva dva, većina voksela prostora vidljiva iz barem jednog od osjetila. Jedno osjetilo razmješteno je u gornjem lijevom kutu prostora te tako pokriva



(a) (b)

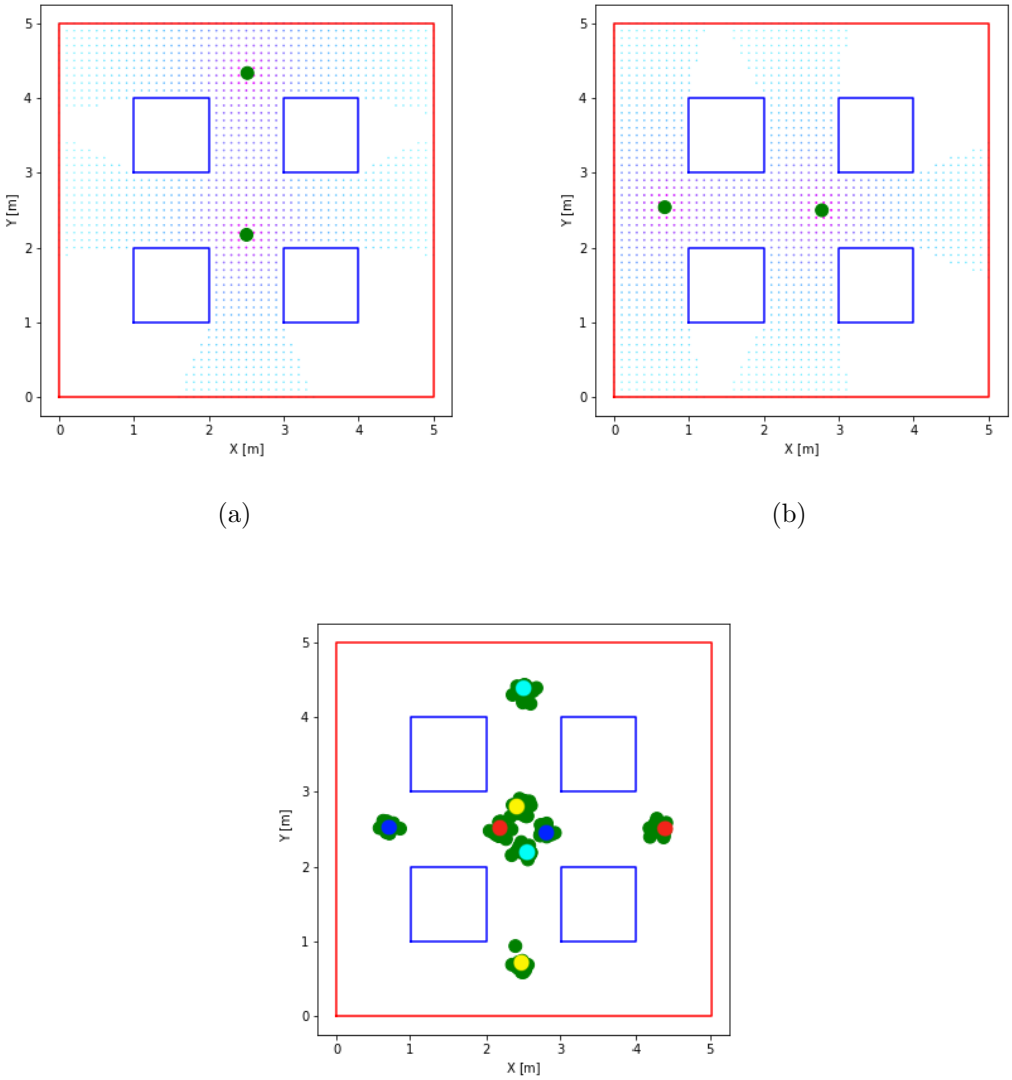
(c)

Slika 8.9: Primjer razmještaja dva usmjerena osjetila u dvodimenzionalnom modelu Ispitnog prostora 1

veći dio prostora, a drugo je razmješteno u dio prostora zaklonjen srednjom pregradom.

Na slici 8.10, prikazan je primjer razmještaja dva svesmjerna osjetila u dvodimenzionalnom modelu Ispitnog prostora 2. Na slikama 8.10 a) i b) prikazana su dva slična razmještaja. Kako je ovaj prostor simetričan oko svoje srednje točke, višestruki razmještaji daju jednake vrijednosti pokrivenosti prostora. Razmještaj parova osjetila za sva ponavljanja ovog ispitnog slučaja dan je na slici 8.10 c). Nekoliko odabranih parova osjetila označeno je različitim bojama, kako bi se prikazala simetrija njihovog razmještaja.

U trećem je primjeru, slika 8.11, ponovno prikazan razmještaj dva svesmjerna osjetila,



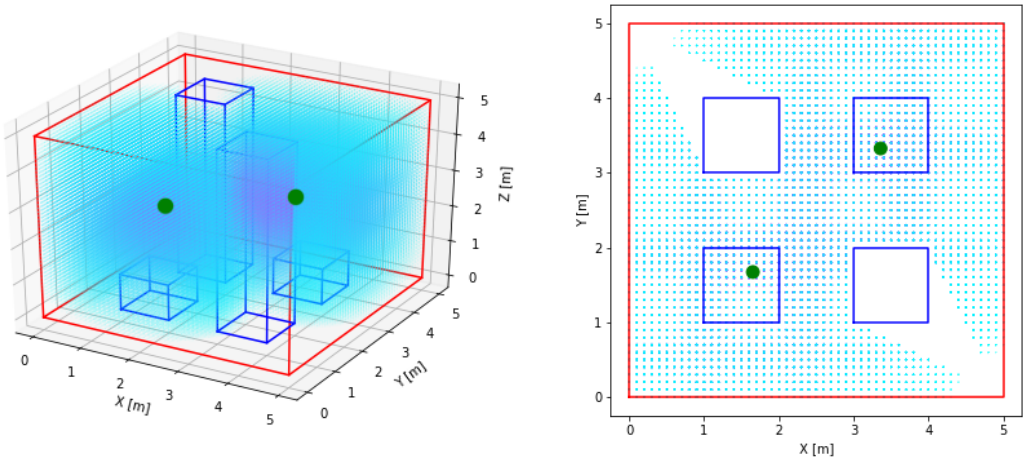
(a) (b)

(c)

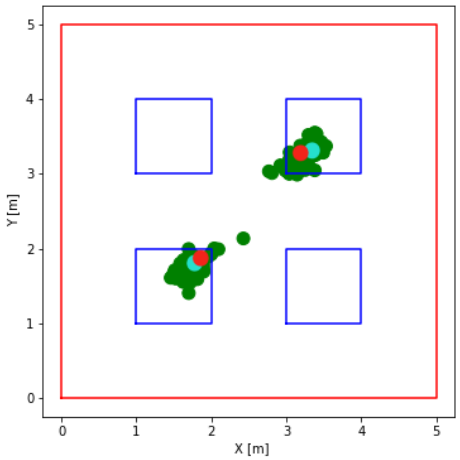
Slika 8.10: Primjer razmještaja dva svesmjerna osjetila u dvodimenzionalnom modelu Ispitnog prostora 2

ali u trodimenzionalnom modelu Ispitnog prostora 2. Na slici 8.11 a) dan je trodimenzionalni prikaz prostora, a na slici 8.11 b) prikaz tlocrta. U ovom su primjeru osjetila razmještena iznad dva niža stupa te tako uspijevaju pokriti i prostor iza njih. Na slici 8.11 c) prikazan je razmještaj svih parova osjetila. Ponovno, nekoliko parova osjetila je naznačeno različitom bojom.

U četvrtom primjeru, slika 8.12, prikazan je razmještaj usmjerenih osjetila u dvodimenzionalnom modelu Ispitnog prostora 3. Na slikama 8.12 a) i b) prikazan je rezultat razmještaja jednog osjetila. Na slici a) dan je primjer razmještaja s najmanjom pokri-



(a) (b)

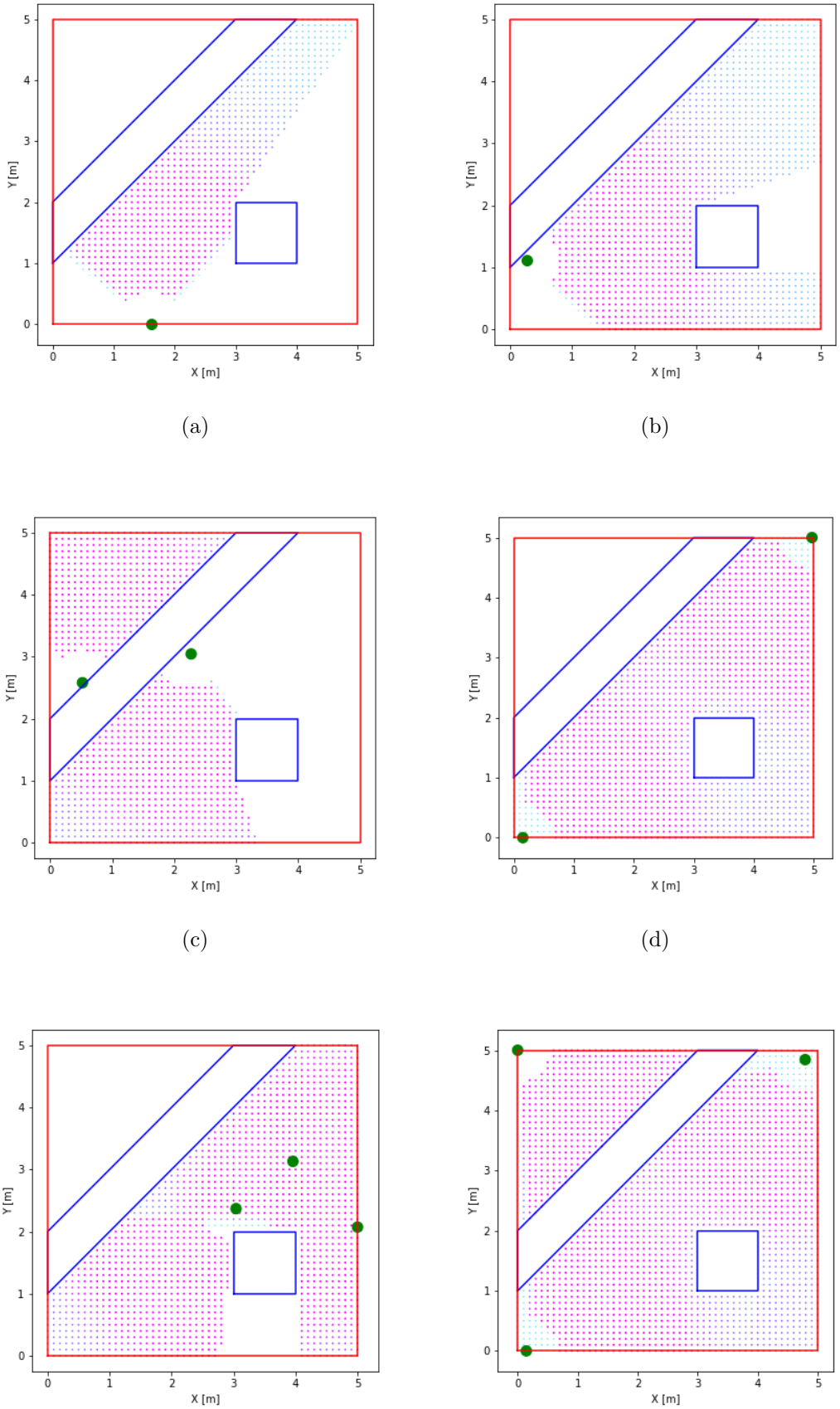


(c)

Slika 8.11: Primjer razmještaja dva svesmjerna osjetila u trodimenzionalnom modelu Ispitnog prostora 2

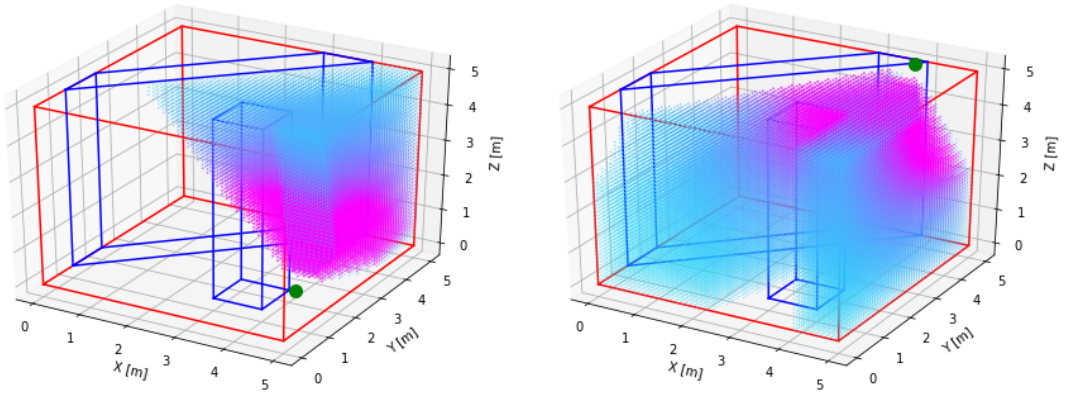
venosti prostora $C = 0{,}2685$, a na slici b) primjer razmještaja s najvećom pokrivenosti $C = 0{,}4108$. U oba slučaja algoritam je odabrao pozicije unutar većeg dijela prostora, ali i male promjene u kutu i pomaku osjetila značajno su utjecale na vrijednost pokrivenosti prostora.

Na slikama 8.12 c) i d) razmještena su dva usmjerena osjetila. Ponovno, na slici c) dan je prikaz razmještaja s najmanjom pokrivenosti prostora $C = 0{,}4842$, a na slici d) s najvećom pokrivenosti prostora $C = 0{,}6504$. U prvom slučaju algoritam je razmjestio osjetila u oba dijela prostora te je ostvario manju pokrivenost prostora. Moguć razlog



Slika 8.12: Primjer razmještaja usmjerenih osjetila u dvodimenzionalnom modelu Ispitnog prostora 3

tome je lošiji razmještaj osjetila u većem dijelu prostora. U najboljem slučaju oba osjetila su razmještena u većem dijelu prostora, jedno nasuprot drugom.



(a) (b)

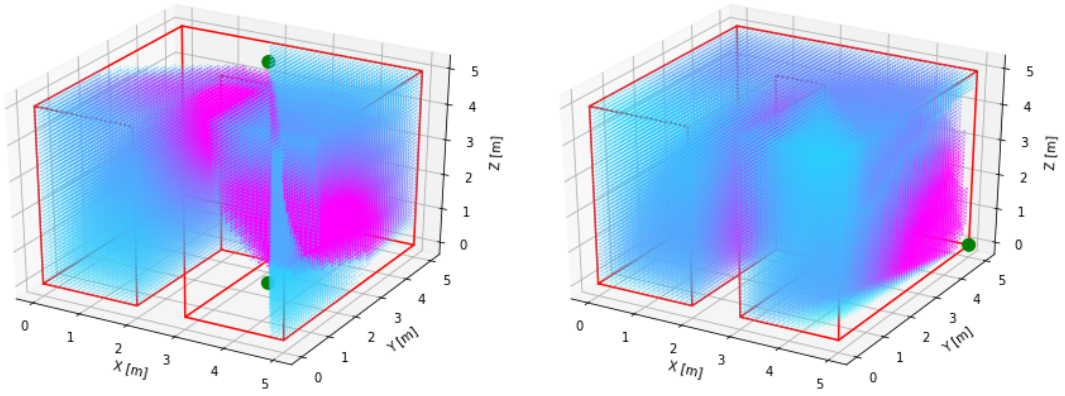
Slika 8.13: Primjer razmještaja usmjerenog osjetila u trodimenzionalnom modelu Ispitnog prostora 3

Razmještaj tri osjetila dan je na slikama 8.12 e) i f). U najgorem slučaju sva tri osjetila raspoređena su u većem dijelu prostora, dosta blizu jedno drugome. Time cijeli dio prostora s lijeve strane pregrade ostaje nepokriven i rezultira najmanjom vrijednosti pokrivenosti prostora $C = 0{,}6651$. Za razmještaj s najvećom pokrivenosti prostora $C = 0{,}8491$ dva osjetila su razmještena u većem dijelu, jedno nasuprot drugom, a treće je razmješteno tako da pokriva manji dio prostora u potpunosti.

U petom primjeru, slika 8.13, prikazan je razmještaj usmjerenog osjetila u trodimenzionalnom modelu Ispitnog prostora 3. Na slici 8.13 a) dan je primjer razmještaja s najmanjom pokrivenosti prostora $C = 0{,}1886$, a na slici b) primjer razmještaja s najvećom pokrivenosti $C = 0{,}2758$.

Na slici 8.14, prikazan je primjer razmještaja dva usmjerena osjetila u trodimenzionalnom modelu Ispitnog prostora 1. Razmještaj s najmanjom pokrivenosti prostora $C = 0{,}4329$ prikazan je na slici 8.14 a), a razmještaj s najvećom pokrivenosti $C = 0{,}5105$ na slici b).

Dodatno, prikazana je i raspršenost odabranih razmještaja dva svesmjerna osjetila u trodimenzionalne modele sva tri ispitna prostora. U tablici 8.5 navedeni su rezultati



(a) (b)

Slika 8.14: Primjer razmještaja dva usmjerena osjetila u trodimenzionalnom modelu Ispitnog prostora 1

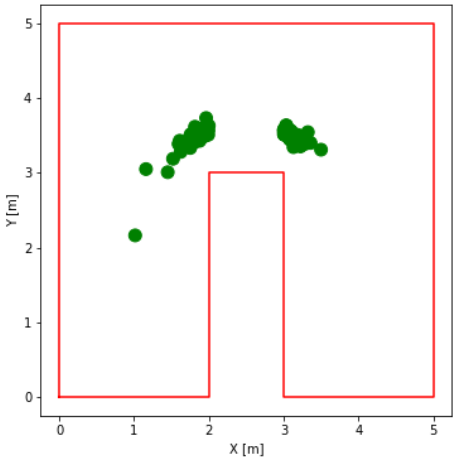
Tablica 8.5: Rezultati pokrivenosti prostora C šest odabranih algoritama za ispitni slučaj s dva svesmjerna osjetila razmještana trodimenzionalne modele sva tri ispitna prostora s rasterizacijskom vrijednosti $r_v = 0{,}1\,m$

	Ispitni prostor 1	Ispitni prostor 2	Ispitni prostor 3
ABC	0.1510	0.1526	0.1422
FWA	0.1369	0.1432	0.1297
PSO	**0.1512**	**0.1533**	**0.1434**
HCA	0.1462	0.1484	0.1370
NMM	0.1270	0.1386	0.1238
SA	0.1321	0.1300	0.1204

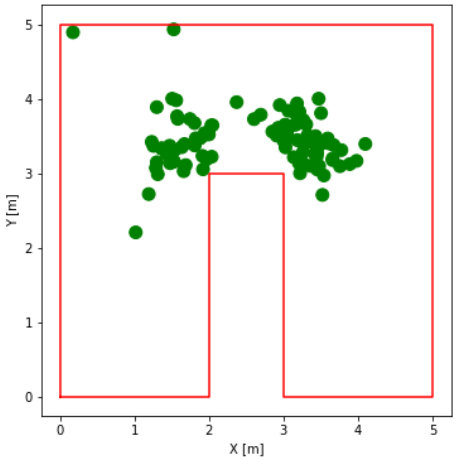
ostvarene pokrivenosti prostora za svaki od algoritama. Iz rezultata je vidljivo da je algoritam optimizacije roja čestica (PSO) ostvario najbolje rezultate za ovaj skup ispitnih slučajeva, dok je algoritam umjetnog roja pčela (ABC) po rezultatima drugi.

Pozicije oba svesmjerna osjetila, odnosno njihov razmještaj, svih 100 ponavljanja prikazani su na tlocrtima sva tri ispitna prostora na slikama 8.15, 8.16 i 8.17. Za sva tri ispitna prostora vidljivo je algoritam optimizacije roja čestica (PSO) ima najmanje raspršenje razmještaja. Algoritmi umjetnog roja pčela (ABC) i uspona uz brijeg (HCA) postižu približno iste razmještaje i u odabiru nemaju puno pozicija koje odstupaju. Prikazani razmještaj ostvaren algoritmom vatrometa (FWA) ima neke odabrane pozicije koje od-

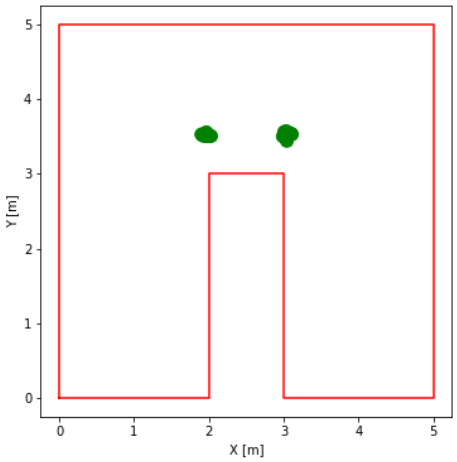
stupaju od očekivanih. Nelder-Mead metoda (NMM) i simulirano kaljenje (SA) pokazuju veliku raspršenost odabranog razmještaja. Prikazane raspršenosti direktno odgovaraju ostvarenim rezultatima iz tablice 8.5, odnosno vidljivo je da algoritmi čiji je razmještaj manje raspršen ostvaruju bolje rezultate.



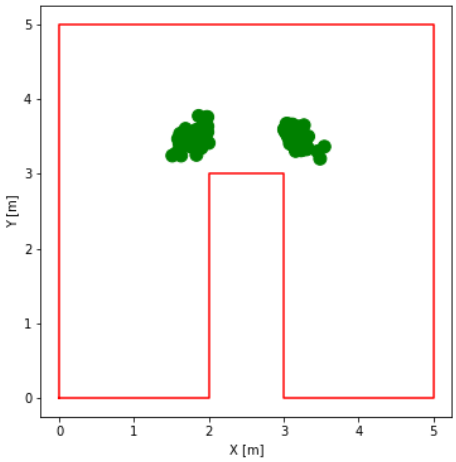
(a) ABC



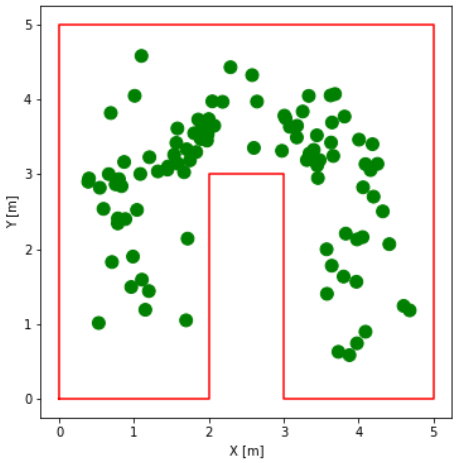
(b) FWA



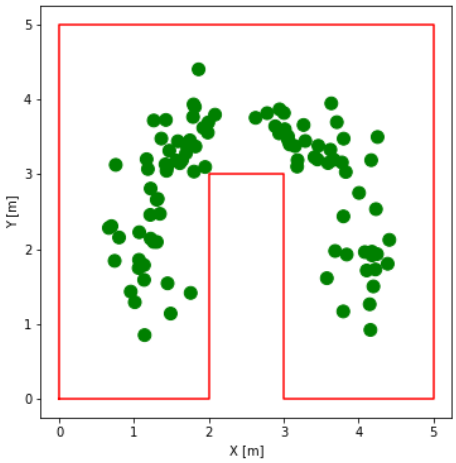
(c) PSO



(d) HCA

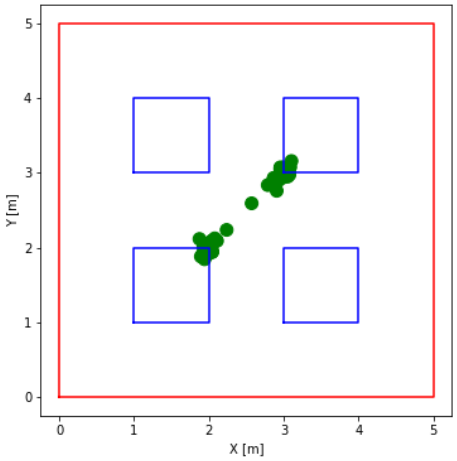


(e) NMM

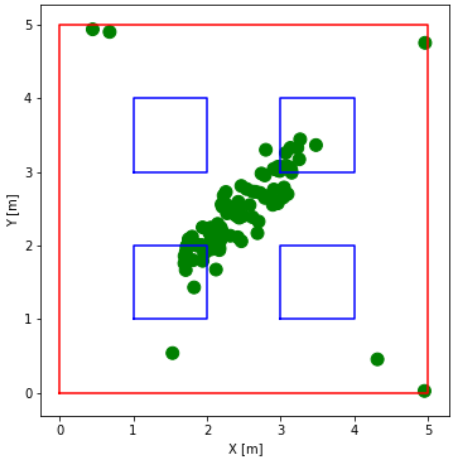


(f) SA

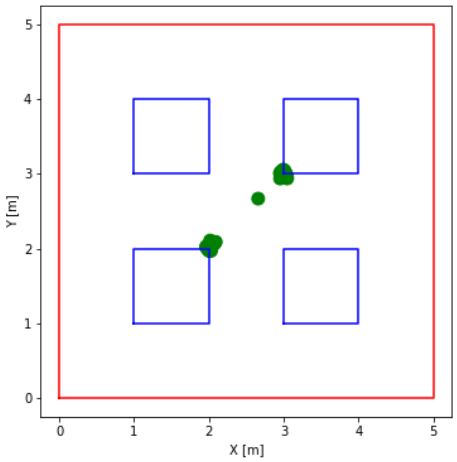
Slika 8.15: Primjer raspršenosti rezultata 100 ponavljanja svakog od algoritama za Ispitni prostor 1



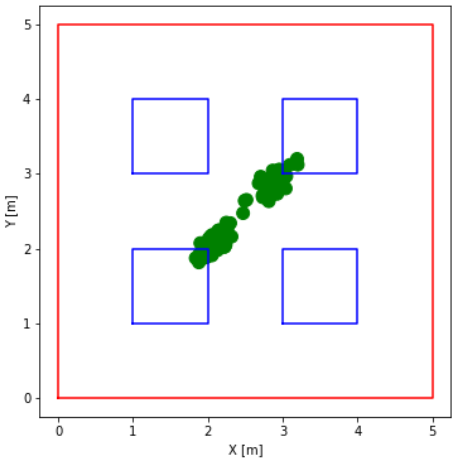
(a) ABC



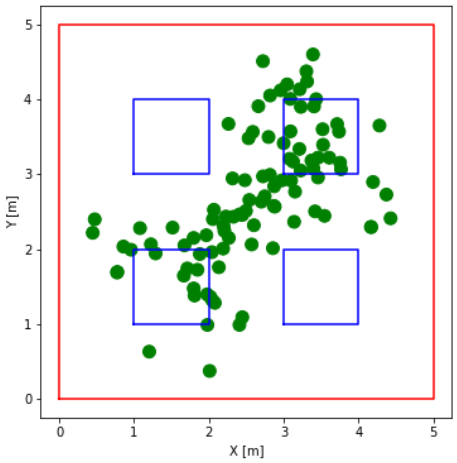
(b) FWA



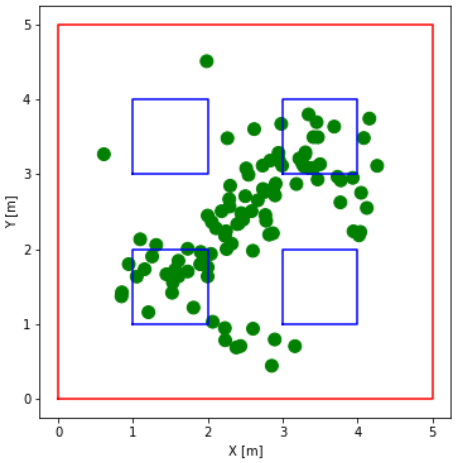
(c) PSO



(d) HCA

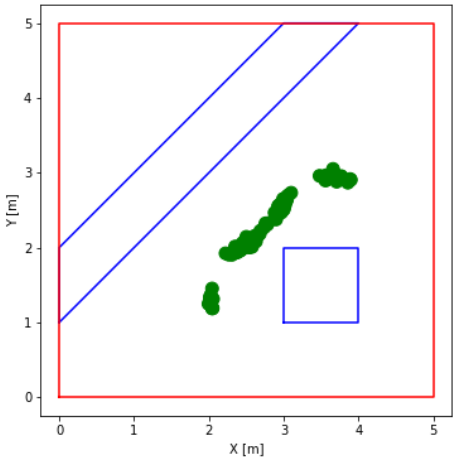


(e) NMM

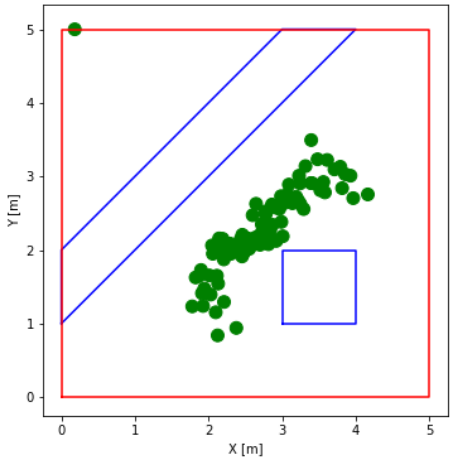


(f) SA

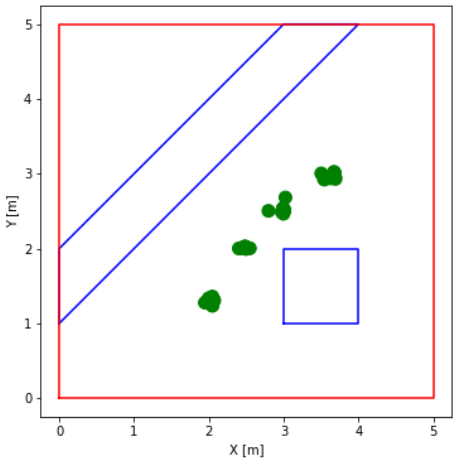
Slika 8.16: Primjer raspršenosti rezultata 100 ponavljanja svakog od algoritama za Ispitni prostor 2



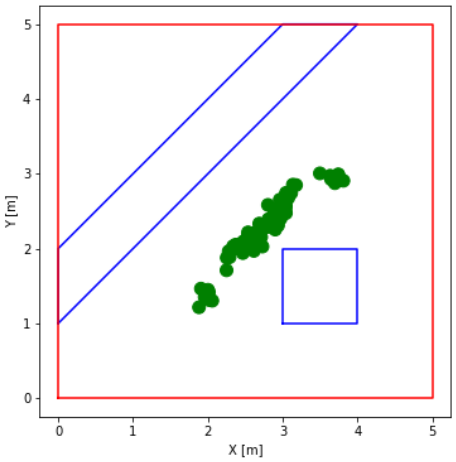
(a) ABC



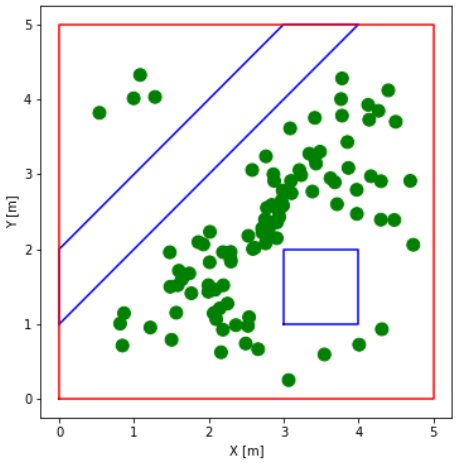
(b) FWA



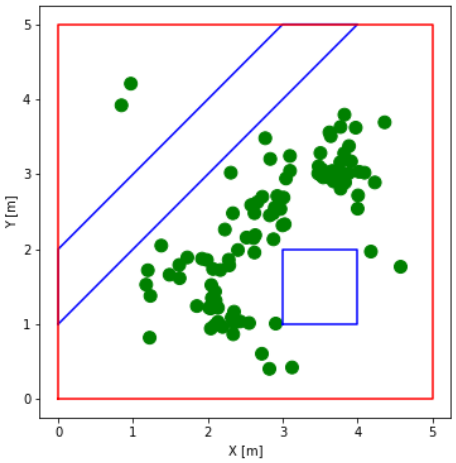
(c) PSO



(d) HCA



(e) NMM



(f) SA

Slika 8.17: Primjer raspršenosti rezultata 100 ponavljanja svakog od algoritama za Ispitni prostor 3

9. Poglavlje

ZAKLJUČAK

U ovom se radu istražio problem razmještaja osjetila u dvodimenzionalnom i trodimenzionalnom zatvorenom prostoru. Predloženi pristup modeliranju prostora omogućava izradu modela za koje je moguće procijeniti njegovu pokrivenost osjetilima. Kako bi razmještaj osjetila odgovarao stvarnoj problematici, predložen je model osjetilnih sposobnosti predstavnika svesmjernih i usmjerenih osjetila. Navedene dvije komponente, modeli prostora i osjetilnih sposobnosti osjetila, združene su u model razmještaja osjetila za pokrivanje zatvorenoga prostora kroz optimizacijsku funkciju temeljenu na metrici pokrivenosti područja.

Rezultati odabranih optimizacijskih algoritama uspoređeni su s rezultatima dobivenim koristeći pristup iscrpnog pretraživanja (ES) te rekurzivnog iscrpnog pretraživanja (RES) za nekoliko ispitnih slučajeva. Relativna pogreška u pokrivenosti prostora za odabrani razmještaj između rekurzivnog iscrpnog pretraživanja (RES) i algoritma optimizacije roja čestica (PSO), stohastičkog algoritma koji je za navedene ispitne slučajeve postigao najbolji rezultat, u prosjeku iznosi 0.14%. Za trodimenzionalni model prostora odabrani stohastički optimizacijski algoritmi ostvarili su ubrzanje od nekoliko desetaka puta u odnosu na rekurzivno iscrpno pretraživanje (RES).

Provedena je i eksperimentalna provjera razmještanja kombinacija dviju vrsta osjetila u dvodimenzionalnim i trodimenzionalnim modelima triju prostora. Estimote *Bluetooth* oznaka niske energije predstavljala je svesmjerna, a Stereolabs ZED kamera usmjerena osjetila. Razmještano je između jedan i tri osjetila svake vrste osjetila. Za svaki od modela svakog od prostora ispitane su tri rasterizacijske vrijednosti. Šest odabranih opti-

mizacijskih algoritama je analizirano i uspoređeno nad velikim brojem ispitnih slučajeva.

Usporedba algoritama provedena je koristeći tri pristupa. Prvi pristup zasnivao se na brojanju pobjeda, odnosno najboljih rezultata u svakoj od kategorija najmanje, srednje i najveće vrijednosti. Drugi pristup usporedbi nadogradio je prethodni s provjerom statističke značajnosti za svaki od ispitnih slučajeva. Za statističku analizu korištena je ANOVA-u ponovljenih mjerenja s Greenhouse-Geisser korekcijom, u slučaju povrede sferičnosti podataka, te naknadnom analizom koristeći Bonferroni korekciju. Treći se pristup zasnivao na statističkoj analizi svih rezultata koristeći istu metodologiju kao u prethodnom pristupu.

Za sva tri pristupa algoritam umjetnog roja pčela (ABC) ostvario je najveću pokrivenost osjetilima. U prva dva pristupa algoritam optimizacije roja čestica (PSO) pokazao se sljedećim najboljim, a algoritam uspona uz brijeg (HCA) odmah nakon njega, posebice u kategoriji najveće vrijednosti. Pri statističkoj analizi svih rezultata, trećem pristupu, pokazalo se da je algoritam uspona uz brijeg (HCA) ipak statistički značajno bolji od algoritma optimizacije roja čestica (PSO). Isti rezultati dobiveni su i nakon statističke analiza podskupa s trodimenzionalnim modelima prostora.

Kroz prva se dva pristupa usporedbi pokazalo da je simulirano kaljenje (SA) najbrži algoritam od ispitanih. U trećem se pristupu pokazalo da ne postoji statistički značajna razlika vremena izvođenja algoritama na cijelom ispitnom skupu niti na podskupu s trodimenzionalnim modelima prostora.

Kako vrijeme izvođenja optimizacijskih algoritama raste linearno s brojem točaka kojima je prostor opisan te je izgledno da će se za prostore s velikim brojem točaka, bilo zbog veličine prostora ili vrijednosti rasterizacije, pojaviti problem većeg vremena izvođenja. U daljnjem istraživanju analizirat će se utjecaj raznih značajki na vrijeme izvođenja optimizacije te dobivene rezultate pokrivenosti prostora osjetilima. Ispitat će se utjecaj veličine rasterizacijske vrijednosti prostora te analizirati mogućnost smanjenja broja voksela u prostoru bez značajnijeg gubitka na točnosti optimizacijskih algoritama. Dodatno će se ispitati i broj izvršavanja optimizacijske funkcije po nezavisnoj varijabli koja se optimizira. Cilj je odabrati značajke koji će omogućiti brže izvođenje algoritama uz zadržavanje približno iste ostvarene pokrivenosti prostora. Bit će ispitani i drugi optimizacijski algoritmi te druge metrike pokrivenosti prostora, posebice k-pokrivenosti. Ideja je dodati nove modele osjetilnih sposobnosti osjetila različitih vrsta i pojednostaviti izradu modela osjetila.

Jedan od mogućih pristupa je korištenje ojačanog učenja (eng. reinforcement learning) koje omogućava treniranje agenata, odnosno osjetila, koje se potom može razmještati u nepoznatim prostorima. Programski paket razvijen kroz ovo istraživanje ponudit će se istraživačkoj zajednici kao jedno od okruženja za usporedbu optimizacijskih algoritama.

LITERATURA

[1] D. Arbula, "Raspodijeljeni algoritam za lokalizaciju u neusidrenoj mreži određivanjem smjera dolaska signala", Diplomski/Magistarski rad, University of Zagreb. Faculty of Electrical Engineering and Computing, 2008.

[2] W. Wu, Z. Zhang, W. Lee, D. Du *i surad.*, *Optimal Coverage in Wireless Sensor Networks*. Springer, 2020.

[3] D. Sušanj, "Raspodijeljeni algoritam za odabir susjeda u grafu mreže uz zadržavanje svojstva krutosti", Diplomski/Magistarski rad, University of Rijeka, Faculty of Engineering, 2015.

[4] V. L. Boginski, C. W. Commander, P. M. Pardalos, i Y. Ye, *Sensors: Theory, Algorithms, and Applications*. Springer Science & Business Media, 2011, svezak 61.

[5] T. O'Donovan, J. O'Donoghue, C. Sreenan, D. Sammon, P. O'Reilly, i K. A. O'Connor, "A context aware wireless body area network (BAN)", u *2009 3rd International Conference on Pervasive Computing Technologies for Healthcare*. IEEE, 2009, str. 1–8.

[6] V. Peiris, "Highly integrated wireless sensing for body area network applications", *SPIE Newsroom*, svezak 10, broj 2.1201312, str. 00512, 2013.

[7] M. Bilal i S.-G. Kang, "An authentication protocol for future sensor networks", *Sensors*, svezak 17, broj 5, str. 979, 2017.

[8] J. K. Hart i K. Martinez, "Environmental sensor networks: A revolution in the earth system science?" *Earth-Science Reviews*, svezak 78, broj 3-4, str. 177–191, 2006.

[9] A. Tiwari, P. Ballal, i F. L. Lewis, "Energy-efficient wireless sensor network design and implementation for condition-based maintenance", *ACM Transactions on Sensor Networks (TOSN)*, svezak 3, broj 1, str. 1–es, 2007.

[10] K. Saleem, N. Fisal, i J. Al-Muhtadi, "Empirical studies of bio-inspired self-organized secure autonomous routing protocol", *IEEE Sensors Journal*, svezak 14, broj 7, str. 2232–2239, 2014.

[11] S. Meguerdichian, F. Koushanfar, M. Potkonjak, i M. B. Srivastava, "Coverage Problems in Wireless Ad-hoc Sensor Networks", u *Proceedings IEEE INFOCOM 2001. Conference on computer communications. Twentieth annual joint conference of the IEEE computer and communications society (Cat. No. 01CH37213)*, svezak 3. IEEE, 2001, str. 1380–1387.

[12] C.-F. Huang, Y.-C. Tseng, i L.-C. Lo, "The Coverage Problem in Three-Dimensional Wireless Sensor Networks", u *IEEE Global Telecommunications Conference, 2004. GLOBECOM'04.*, svezak 5. IEEE, 2004, str. 3182–3186.

[13] C.-F. Huang i Y.-C. Tseng, "The Coverage Problem in a Wireless Sensor Network", *Mobile networks and Applications*, svezak 10, broj 4, str. 519–528, 2005.

[14] M. Liu, J. Cao, W. Lou, L.-j. Chen, i X. Li, "Coverage Analysis for Wireless Sensor Networks", u *International Conference on Mobile Ad-Hoc and Sensor Networks*. Springer, 2005, str. 711–720.

[15] D. Arbula, "Raspodijeljeni algoritam za određivanje položaja čvorova u neusidrenoj bežićnoj mreži osjetila", Doktorska disertacija, University of Zagreb. Faculty of Electrical Engineering and Computing, 2014.

[16] G. Zhang, B. Dong, i J. Zheng, "Visual Sensor Placement and Orientation Optimization for Surveillance Systems", u *2015 10th International Conference on Broadband and Wireless Computing, Communication and Applications (BWCCA)*. IEEE, 2015, str. 1–5.

[17] P. Santi, *Topology Control in Wireless Ad Hoc and Sensor Networks*. John Wiley & Sons, 2005.

[18] R. Bodor, A. Drenner, P. Schrater, i N. Papanikolopoulos, "Optimal Camera Placement for Automated Surveillance Tasks", *Journal of Intelligent and Robotic Systems*, svezak 50, broj 3, str. 257–295, 2007.

[19] J. Kritter, M. Brévilliers, J. Lepagnot, i L. Idoumghar, "On the optimal placement of cameras for surveillance and the underlying set cover problem", *Applied Soft Computing*, svezak 74, str. 133–153, 2019.

[20] J. Kritter, "On the optimal placement of cameras for the surveillance of urban events: a real-world, human-assisted combinatorial approach for decision support systems", Doktorska disertacija, Université de Haute-Alsace, 2020.

[21] R. Honsberger, *Mathematical Plums*. Mathematical Association of America, 1979.

[22] V. Chvátal, "A Combinatorial Theorem in Plane Geometry", *Journal of Combinatorial Theory, Series B*, svezak 18, broj 1, str. 39–41, 1975.

[23] M. De Berg, M. Van Kreveld, M. Overmars, i O. Schwarzkopf, "Computational geometry", u *Computational geometry*. Springer, 1997, str. 1–17.

[24] B. J. Nilsson, "Guarding Art Galleries: Methods for Mobile Guards", Doktorska disertacija, Department of Computer Science; Lund University, 1996.

[25] H. González-Banos, "A Randomized Art-Gallery Algorithm for Sensor Placement", u *Proceedings of the seventeenth annual symposium on Computational geometry*, 2001, str. 232–240.

[26] S. Fisk, "A short proof of Chvátal's Watchman Theorem", *Journal of Combinatorial Theory, Series B*, svezak 24, broj 3, str. 374, 1978.

[27] J. O'Rourke, *Art Gallery Theorems and Algorithms*. Oxford University Press Oxford, 1987, svezak 57.

[28] T. C. Shermer, "Recent Results in Art Galleries", *Proceedings of the IEEE*, svezak 80, broj 9, str. 1384–1399, 1992.

[29] J. Urrutia, "Art Gallery and Illumination Problems", u *Handbook of computational geometry*. Elsevier, 2000, str. 973–1027.

[30] J. Marzal, "The three-dimensional art gallery problem and its solutions", Doktorska disertacija, School of Information Technology, Murdoch University, 2012.

[31] M. Marengoni, B. A. Draper, A. Hanson, i R. Sitaraman, "A system to place observers on a polyhedral terrain in polynomial time", *Image and Vision Computing*, svezak 18, broj 10, str. 773–780, 2000.

[32] G. Viglietta, "Guarding and Searching Polyhedra", *arXiv preprint arXiv:1211.2483*, 2012.

[33] R. M. Karp, "Reducibility among combinatorial problems", u *Complexity of computer computations*. Springer, 1972, str. 85–103.

[34] W. J. Masek, "Some NP-complete set covering problems", *Unpublished manuscript*, 1979.

[35] X. Li, W. Yu, X. Lin, i S. S. Iyengar, "On Optimizing Autonomous Pipeline Inspection", *IEEE Transactions on Robotics*, svezak 28, broj 1, str. 223–233, 2012.

[36] G. Safak, *The art-gallery problem: A survey and an extension*. Skolan för datavetenskap och kommunikation, Kungliga Tekniska högskolan, 2009.

[37] N. Chesnokov, "The Art Gallery Problem: An Overview and Extension to Chromatic Coloring and Mobile Guards", 2018.

[38] V. Akbarzadeh, A. H.-R. Ko, C. Gagné, i M. Parizeau, "Topography-Aware Sensor Deployment Optimization with CMA-ES", u *International Conference on Parallel Problem Solving from Nature*. Berlin, Heidelberg: Springer, 2010, str. 141–150.

[39] A. A. Altahir, V. S. Asirvadam, N. H. B. Hamid, P. Sebastian, N. B. Saad, R. B. Ibrahim, i S. C. Dass, "Optimizing Visual Sensor Coverage Overlaps for Multiview Surveillance Systems", *IEEE Sensors Journal*, svezak 18, broj 11, str. 4544–4552, 2018.

[40] M. Hefeeda i H. Ahmadi, "Energy Efficient Protocol for Deterministic and Probabilistic Coverage in Sensor Networks", *IEEE Transactions on Parallel and Distributed Systems*, svezak 21, broj 5, str. 579–593, 2009.

[41] N. Ahmed, S. S. Kanhere, i S. Jha, "Probabilistic coverage in wireless sensor networks", u *The IEEE Conference on Local Computer Networks 30th Anniversary (LCN'05) l.* IEEE, 2005, str. 8–pp.

[42] Y. Zou i K. Chakrabarty, "Sensor Deployment and Target Localization in Distributed Sensor Networks", *ACM Transactions on Embedded Computing Systems (TECS)*, svezak 3, broj 1, str. 61–91, 2004.

[43] Y. Zou i K. Chakrabarty, "A Distributed Coverage- and Connectivity-Centric Technique for Selecting Active Nodes in Wireless Sensor Networks", *IEEE Transactions on Computers*, svezak 54, broj 8, str. 978–991, 2005.

[44] B. Liu i D. Towsley, "A Study on the Coverage of Large-Scale Sensor Networks", u *2004 IEEE international conference on mobile ad-hoc and sensor systems (IEEE Cat. No. 04EX975).* IEEE, 2004, str. 475–483.

[45] P. Hall i P. G. Hall, *Introduction to the theory of coverage processes.* John Wiley & Sons Incorporated, 1988, svezak 201.

[46] H. Zhang, J. C. Hou *i surad.*, "Maintaining Sensing Coverage and Connectivity in Large Sensor Networks", *Ad Hoc Sens. Wirel. Networks*, svezak 1, broj 1-2, str. 89–124, 2005.

[47] L. Wu, H. Du, W. Wu, D. Li, J. Lv, i W. Lee, "Approximations for Minimum Connected Sensor Cover", u *2013 Proceedings IEEE INFOCOM.* IEEE, 2013, str. 1187–1194.

[48] Y. L. Du i L. Wu, "How Many Target Points Can Replace a Target Area?" u *2014 10th International Conference on Mobile Ad-hoc and Sensor Networks.* IEEE, 2014, str. 120–122.

[49] M. Cardei i J. Wu, "Energy-efficient coverage problems in wireless ad-hoc sensor networks", *Computer communications*, svezak 29, broj 4, str. 413–420, 2006.

[50] S. N. Alam i Z. J. Haas, "Coverage and Connectivity in Three-Dimensional Networks", u *Proceedings of the 12th annual international conference on Mobile computing and networking*, 2006, str. 346–357.

[51] V. Akbarzadeh, J.-C. Lévesque, C. Gagné, i M. Parizeau, "Efficient Sensor Placement Optimization Using Gradient Descent and Probabilistic Coverage", *Sensors*, svezak 14, broj 8, str. 15 525–15 552, 2014.

[52] M. Krishnan, V. Rajagopal, i S. Rathinasamy, "Performance evaluation of sensor deployment using optimization techniques and scheduling approach for K-coverage in WSNs", *Wireless Networks*, svezak 24, broj 3, str. 683–693, 2018.

[53] S. S. Dhillon i K. Chakrabarty, "Sensor Placement for Effective Coverage and Surveillance in Distributed Sensor Networks", u *2003 IEEE Wireless Communications and Networking, 2003. WCNC 2003.*, svezak 3. IEEE, 2003, str. 1609–1614.

[54] E. Hörster i R. Lienhart, "Approximating optimal visual sensor placement", u *2006 IEEE International Conference on Multimedia and Expo*. IEEE, 2006, str. 1257–1260.

[55] E. Hörster i R. Lienhart, "On the Optimal Placement of Multiple Visual Sensors", u *Proceedings of the 4th ACM International Workshop on Video Surveillance and Sensor Networks*, ser. VSSN '06. New York, NY, USA: Association for Computing Machinery, 2006, str. 111–120. [Online]. Dostupno na: https://doi.org/10.1145/1178782.1178800

[56] M. Pålsson i J. Ståhl, *The Camera Placement Problem - An art gallery problem variation*. Department of Computer Science, Lund University, 2008.

[57] J.-J. Gonzalez-Barbosa, T. García-Ramírez, J. Salas, J.-B. Hurtado-Ramos *i surad.*, "Optimal Camera Placement for Total Coverage", u *2009 IEEE International Conference on Robotics and Automation*. IEEE, 2009, str. 844–848.

[58] S. M. A. Salehizadeh, A. Dirafzoon, M. B. Menhaj, i A. Afshar, "Coverage in Wireless Sensor Networks Based on Individual Particle Optimization", u *2010 International Conference on Networking, Sensing and Control (ICNSC)*. IEEE, 2010, str. 501–506.

[59] Y. Morsly, N. Aouf, M. S. Djouadi, i M. Richardson, "Particle Swarm Optimization Inspired Probability Algorithm for Optimal Camera Network Placement", *IEEE Sensors Journal*, svezak 12, broj 5, str. 1402–1412, 2011.

[60] V. Akbarzadeh, C. Gagné, M. Parizeau, i M. A. Mostafavi, "Black-box Optimization of Sensor Placement with Elevation Maps and Probabilistic Sensing Models", u *2011 IEEE International Symposium on Robotic and Sensors Environments (ROSE)*. IEEE, 2011, str. 89–94.

[61] V. Akbarzadeh, C. Gagne, M. Parizeau, M. Argany, i M. A. Mostafavi, "Probabilistic Sensing Model for Sensor Placement Optimization based on Line-of-sight Coverage", *IEEE Transactions on Instrumentation and Measurement*, svezak 62, broj 2, str. 293–303, 2012.

[62] A. A. Altahir, V. S. Asirvadam, N. H. B. Hamid, i P. Sebastian, "Modeling Camera Coverage Using Imagery Techniques for Surveillance Applications", u *2014 IEEE International Conference on Control System, Computing and Engineering (ICCSCE 2014)*. IEEE, 2014, str. 40–45.

[63] A. A. Altahir, V. S. Asirvadam, N. H. B. Hamid, i P. Sebastian, "Optimizing Visual Sensor Parameters for Total Coverage Maximization", u *2014 IEEE Student Conference on Research and Development*. IEEE, 2014, str. 1–6.

[64] A. A. Altahir, V. S. Asirvadam, N. H. Hamid, P. Sebastian, N. Saad, R. Ibrahim, i S. C. Dass, "Modeling Multicamera Coverage for Placement Optimization", *IEEE sensors letters*, svezak 1, broj 6, str. 1–4, 2017.

[65] M. Thiene, Z. S. Khodaei, i M. Aliabadi, "Optimal Sensor Placement for Maximum Area Coverage (MAC) for Damage Localization in Composite Structures", *Smart materials and structures*, svezak 25, broj 9, str. 095037, 2016.

[66] H. T. T. Binh, N. T. Hanh, N. Dey *i surad.*, "Improved Cuckoo Search and Chaotic Flower Pollination optimization algorithm for maximizing area coverage in Wireless Sensor Networks", *Neural computing and applications*, svezak 30, broj 7, str. 2305–2317, 2018.

[67] Z. Liu, W. Jia, i G. Wang, "Area coverage estimation model for directional sensor networks", *International Journal of Embedded Systems*, svezak 10, broj 1, str. 13–21, 2018.

[68] F. Hržić, D. Sušanj, i K. Lenac, "Optimal beacon positioning for indoor drone navigation", u *12th Annual Baška GNSS Conference (2018)*, 2018, str. 109–117.

[69] D. Sušanj, D. Pinčić, i K. Lenac, "Effective Area Coverage of 2D and 3D Environments With Directional and Isotropic Sensors", *IEEE Access*, svezak 8, str. 185 595–185 608, 2020.

[70] E. Haines, "Point in Polygon Strategies", *Graphics Gems*, svezak 4, str. 24–46, 1994.

[71] T. S. Rappaport *i surad.*, *Wireless Communications: Principles and Practice*. Prentice Hall PTR New Jersey, 1996, svezak 2.

[72] S. Mattoccia. Stereo Vision: Algorithms and Applications. Pristupljeno 16.2.2021. [Online]. Dostupno na: http://vision.deis.unibo.it/~smatt/stereo.htm

[73] D. B. Gennery, "Stereo-Camera Calibration", u *Proceedings ARPA IUS Workshop*, 1979, str. 101–107.

[74] R. Hartley i A. Zisserman, "Multiple View Geometry", 1995.

[75] D. V. Papadimitriou i T. J. Dennis, "Epipolar Line Estimation and Rectification for Stereo Image Pairs", *IEEE transactions on image processing*, svezak 5, broj 4, str. 672–676, 1996.

[76] G. Medioni i R. Nevatia, "Segment-based Stereo Matching", *Computer Vision, Graphics, and Image Processing*, svezak 31, broj 1, str. 2–18, 1985.

[77] The JSON data interchange syntax. Pristupljeno 18.5.2021. [Online]. Dostupno na: https://www.ecma-international.org/publications-and-standards/standards/ecma-404/

[78] Slurm Workload Manager. Pristupljeno 18.5.2021. [Online]. Dostupno na: https://slurm.schedmd.com/overview.html

[79] Superračunalo „Bura", Sveučilište u Rijeci, Centar za napredno računanje i modeliranje. Pristupljeno 8.9.2020. [Online]. Dostupno na: https://cnrm.uniri.hr/hr/bura/

[80] Estimote. Pristupljeno 24.12.2020. [Online]. Dostupno na: https://estimote.com/

[81] StereoLabs ZED. Pristupljeno 11.1.2021. [Online]. Dostupno na: https://www.stereolabs.com/zed/

[82] R. E. Dorsey i W. J. Mayer, "Genetic algorithms for estimation problems with multiple optima, nondifferentiability, and other irregular features", *Journal of Business & Economic Statistics*, svezak 13, broj 1, str. 53–66, 1995.

[83] L. A. Machowski i T. Marwala, "Evolutionary Optimisation Methods for Template Based Image Registration ", *arXiv preprint arXiv:0705.1674*, 2007.

[84] C.-T. Chang *i surad.*, "Heuristic optimization methods for three matrix problems", Doktorska disertacija, Catholic University of Louvain, Louvain-la-Neuve, Belgium, 2012.

[85] J. A. Nelder i R. Mead, "A simplex method for function minimization", *The computer journal*, svezak 7, broj 4, str. 308–313, 1965.

[86] S. M. Goldfeld, R. E. Quandt, i H. F. Trotter, "Maximization by Quadratic Hill-Climbing", *Econometrica: Journal of the Econometric Society*, str. 541–551, 1966.

[87] M. Pincus, "A Monte Carlo Method for the Approximate Solution of Certain Types of Constrained Optimization Problems", *Operations research*, svezak 18, broj 6, str. 1225–1228, 1970.

[88] S. Kirkpatrick, C. D. Gelatt, i M. P. Vecchi, "Optimization by Simulated Annealing", *science*, svezak 220, broj 4598, str. 671–680, 1983.

[89] Y. Tan, *Handbook of Research on Design, Control, and Modeling of Swarm Robotics*. IGI Global, 2015.

[90] S. Aslan i D. Karaboga, "A genetic Artificial Bee Colony algorithm for signal reconstruction based big data optimization", *Applied Soft Computing*, svezak 88, str. 106053, 2020.

[91] Z. Yue, S. Zhang, i W. Xiao, "A Novel Hybrid Algorithm Based on Grey Wolf Optimizer and Fireworks Algorithm", *Sensors*, svezak 20, broj 7, str. 2147, 2020.

[92] F. Fausto, A. Reyna-Orta, E. Cuevas, Á. G. Andrade, i M. Perez-Cisneros, "From ants to whales: metaheuristics for all tastes", *Artificial Intelligence Review*, svezak 53, broj 1, str. 753–810, 2020.

[93] J. Kennedy i R. Eberhart, "Particle Swarm Optimization", u *Proceedings of ICNN'95-International Conference on Neural Networks*, svezak 4. IEEE, 1995, str. 1942–1948.

[94] D. Karaboga i B. Basturk, "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm", *Journal of global optimization*, svezak 39, broj 3, str. 459–471, 2007.

[95] Y. Tan i Y. Zhu, "Fireworks Algorithm for Optimization", u *International conference in swarm intelligence*. Berlin, Heidelberg: Springer, 2010, str. 355–364.

[96] J. Li i Y. Tan, "The bare bones fireworks algorithm: A minimalist global optimizer", *Applied Soft Computing*, svezak 62, str. 454–462, 2018.

[97] A. Prügel-Bennett, "Benefits of a population: Five mechanisms that advantage population-based algorithms", *IEEE Transactions on Evolutionary Computation*, svezak 14, broj 4, str. 500–517, 2010.

[98] E.-G. Talbi, *Metaheuristics: from design to implementation*. John Wiley & Sons, 2009, svezak 74.

[99] G. Mauša, T. G. Grbac, B. D. Bašić, i M.-O. Pavčević, "Hill Climbing and Simulated Annealing in Large Scale Next Release Problem", u *Eurocon 2013*. IEEE, 2013, str. 452–459.

[100] V. Beiranvand, W. Hare, i Y. Lucet, "Best practices for comparing optimization algorithms", *Optimization and Engineering*, svezak 18, broj 4, str. 815–848, 2017.

[101] K. Hussain, M. N. M. Salleh, S. Cheng, i Y. Shi, "On the exploration and exploitation in popular swarm-based metaheuristic algorithms", *Neural Computing and Applications*, svezak 31, broj 11, str. 7665–7683, 2019.

[102] A. Sharma i R. Rani, "KSRMF: Kernelized similarity based regularized matrix factorization framework for predicting anti-cancer drug responses", *Journal of Intelligent & Fuzzy Systems*, svezak 35, broj 2, str. 1779–1790, 2018.

[103] S. P. Lim i H. Haron, "Performance comparison of genetic algorithm, differential evolution and particle swarm optimization towards benchmark functions", u *2013 IEEE Conference on Open Systems (ICOS)*. IEEE, 2013, str. 41–46.

POPIS SLIKA

POPIS TABLICA

POPIS ALGORITAMA I IZVORIŠNIH KODOVA

PRILOZI

A. Izvorišni kodovi

Izvorišni kod A.1: Klasa Point

```
class Point():
    def __init__(self, coords=None):
        try:
            self.__x = coords[0]
        except (TypeError, IndexError):
            self.__x = None

        try:
            self.__y = coords[1]
        except (TypeError, IndexError):
            self.__y = None

        try:
            self.__z = coords[2]
        except (TypeError, IndexError):
            self.__z = None

    def __str__(self):
        s = ""

        if self.__x:
            s += str(self.__x)

        if self.__y:
            s += ", " + str(self.__y)

        if self.__z:
            s += ", " + str(self.__z)

        return s

    @property
    def x(self):
        return self.__x

    @property
    def y(self):
        return self.__y

    @property
    def z(self):
        return self.__z

    @property
    def xy(self):
        return [self.__x, self.__y]

    @property
    def xyz(self):
        return [self.__x, self.__y, self.__z]

    @x.setter
    def x(self, value):
        self.__x = value

    @y.setter
    def y(self, value):
        self.__y = value

    @z.setter
    def z(self, value):
        self.__z = value

    @xy.setter
    def xy(self, values):
        self.__x = values[0]
```

```
        self.__y = values[1]

    @xyz.setter
    def xyz(self, values):
        self.__x = values[0]
        self.__y = values[1]
        self.__z = values[2]
```

Izvorišni kod A.2: Klasa Voxel

```
from SensorPositioning.geometry.point import Point


class Voxel(Point):
    def __init__(self, coords=None):
        super().__init__(coords)

        self.__visibility = 0

    @property
    def visibility(self):
        return self.__visibility

    @property
    def loss(self):
        return 1 - self.__visibility

    @property
    def xyzv(self):
        return [0 if x is None else x for x in self.xyz] + [self.
                __visibility]

    @visibility.setter
    def visibility(self, value):
        self.__visibility = value

    @loss.setter
    def loss(self, value):
        self.__visibility = 1 - value
```

Izvorišni kod A.3: Klasa Polygon

```
from collections import MutableSequence
from shapely.geometry import Point as spPoint
from shapely.geometry import Polygon as spPolygon
from shapely.geometry import LineString as spLine
from shapely.ops import split as spSplit


class Polygon(MutableSequence):
    def __init__(self, polygon):
        if not isinstance(polygon, list):
            raise TypeError
        elif len(polygon) < 3:
            raise ValueError

        self.__polygon = polygon

        if self.__polygon[0] == self.__polygon[-1]:
            self.__polygon.pop()

        self.__shapely_polygon = spPolygon(self.__polygon)

    def __len__(self):
        return len(self.__polygon)

    def __delitem__(self, index):
        self.__polygon.__delitem__(index)

    def __getitem__(self, index):
```

```
        return self.__polygon.__getitem__(index)

    def __setitem__(self, index, value):
        if not isinstance(value, tuple):
            raise TypeError
        elif len(value) < len(self.__polygon[index]):
            raise ValueError

        self.__polygon.__setitem__(index, value)

    def insert(self, index, value):
        if not isinstance(value, tuple):
            raise TypeError
        elif len(value) < len(self.__polygon[0]):
            raise ValueError

        self.__polygon.insert(index, value)

    def append(self, value):
        self.insert(len(self), value)

    def is_point_inside(self, p):
        pt = spPoint(p.xy)
        return self.__shapely_polygon.contains(pt) or self.
                __shapely_polygon.intersects(pt)

    def line_crossings(self, p1, p2):
        line = spLine((spPoint(p1.xy), spPoint(p2.xy)))

        if line.crosses(self.__shapely_polygon):
            crossings = [x.coords[1] for x in spSplit(line, self.
                    __shapely_polygon)]
            return crossings[:-1]
        else:
            return []

    @property
    def x(self):
        try:
            return list(map(lambda x: x[0], self.__polygon))
        except IndexError:
            return None

    @property
    def y(self):
        try:
            return list(map(lambda x: x[1], self.__polygon))
        except IndexError:
            return None

    @property
    def z(self):
        try:
            return list(map(lambda x: x[2], self.__polygon))
        except IndexError:
            return None
```

Izvorišni kod A.4: Klasa Mesh

```
from SensorPositioning.geometry.polygon import Polygon


class Mesh(Polygon):
    def __init__(self, polygon, z_coords=None):
        super().__init__(polygon)

        try:
            self.__bottom = z_coords[0]
            self.__top = z_coords[1]
        except (TypeError, IndexError):
            self.__bottom = None
            self.__top = None
```

```
    def is_point_inside(self, p):
        if self.__bottom is not None and self.__top is not None:
            return super().is_point_inside(p) and self.__bottom <= p.z <
                = self.__top
        else:
            return super().is_point_inside(p)

    def is_line_crossing(self, p1, p2):
        if (self.is_point_inside(p1) and not self.is_point_inside(p2))
                or (not self.is_point_inside(p1) and self.is_point_inside
                (p2)):
            return True

        try:
            crossings = super().line_crossings(p1, p2)
        except (TypeError, ValueError):
            return True

        if crossings != []:
            if self.__bottom is not None and self.__top is not None:
                for c in crossings:
                    if p1.x != p2.x:
                        z = p1.z + (p2.z - p1.z) * ((c[0] - p1.x) / (p2.
                            x - p1.x))
                    elif p1.y != p2.y:
                        z = p1.z + (p2.z - p1.z) * ((c[1] - p1.y) / (p2.
                            y - p1.y))

                    if z >= self.__bottom and z <= self.__top:
                        return True
            else:
                return True

        if self.__bottom is not None and self.__top is not None and
                super().is_point_inside(p1) and super().is_point_inside(
                p2):
            if p1.z > self.__top and p2.z < self.__bottom or p1.z < self
                    .__bottom and p2.z > self.__top:
                return True

        return False

    @property
    def bounds(self):
        if self.__bottom is not None and self.__top is not None:
            return min(self.x), max(self.x), min(self.y), max(self.y),
                    self.__bottom, self.__top
        else:
            return min(self.x), max(self.x), min(self.y), max(self.y)

    @property
    def coords(self):
        if self.__bottom is not None and self.__top is not None:
            return self.x, self.y, [self.__top, self.__bottom]
        else:
            return self.x, self.y
```

Izvorišni kod A.5: Klasa Environment

```
import numpy as np
from SensorPositioning.geometry.voxel import Voxel


class Environment():
    def __init__(self, voxel_size=None):
        self.__positive_meshes = []
        self.__negative_meshes = []

        self.__voxel_size = None
        self.voxel_size = voxel_size
```

```
        self.__environment = []

    def add_positive_mesh(self, mesh):
        self.__positive_meshes.append(mesh)

    def add_negative_mesh(self, mesh):
        self.__negative_meshes.append(mesh)

    def create_environment(self):
        bounds = self.bounds

        if len(bounds) == 4:
            x_min, x_max, y_min, y_max = bounds
            z_min = None
            z_max = None
        elif len(bounds) == 6:
            x_min, x_max, y_min, y_max, z_min, z_max = bounds
        else:
            return

        x_coords = np.linspace(x_min, x_max, int((x_max - x_min) / self.
                __voxel_size + 1))
        y_coords = np.linspace(y_min, y_max, int((y_max - y_min) / self.
                __voxel_size + 1))
        if z_min is None and z_max is None:
            z_coords = [None]
        else:
            z_coords = np.linspace(z_min, z_max, int((z_max - z_min)//
                    self.__voxel_size + 1))

        for x in x_coords:
            for y in y_coords:
                for z in z_coords:
                    v = Voxel((x, y, z))

                    if self.__is_point_valid(v):
                        self.__environment.append(v)

    def __is_point_valid(self, p):
        inside_flag = False

        for m in self.__positive_meshes:
            if m.is_point_inside(p):
                inside_flag = True

        if not inside_flag:
            return False

        for m in self.__negative_meshes:
            if m.is_point_inside(p):
                return False

        return True

    def line_of_sight(self, p1, p2):
        for m in self.__positive_meshes:
            if m.is_line_crossing(p1, p2):
                return False

        for m in self.__negative_meshes:
            if m.is_line_crossing(p1, p2):
                return False

        return True

    @property
    def bounds(self):
        x_min = min([e.bounds[0] for e in self.__positive_meshes])
        x_max = max([e.bounds[1] for e in self.__positive_meshes])
        y_min = min([e.bounds[2] for e in self.__positive_meshes])
        y_max = max([e.bounds[3] for e in self.__positive_meshes])
        try:
            z_min = min([e.bounds[4] for e in self.__positive_meshes])
            z_max = max([e.bounds[5] for e in self.__positive_meshes])
```

```
            return x_min, x_max, y_min, y_max, z_min, z_max
        except IndexError:
            return x_min, x_max, y_min, y_max

    @property
    def lower_bounds(self):
        return self.bounds[::2]

    @property
    def upper_bounds(self):
        return self.bounds[1::2]

    @property
    def positive_meshes(self):
        return self.__positive_meshes

    @property
    def negative_meshes(self):
        return self.__negative_meshes

    @property
    def voxel_size(self):
        return self.__voxel_size

    @property
    def voxel_number(self):
        return len(self.__environment)

    @property
    def voxels(self):
        return self.__environment

    @voxel_size.setter
    def voxel_size(self, value):
        if value is not None:
            self.__voxel_size = value
        else:
            self.__voxel_size = 1
```

Izvorišni kod A.6: Klasa BaseSensor

```
import numpy as np

from SensorPositioning.geometry.point import Point


class BaseSensor(Point):
    def __init__(self, position=None):
        super().__init__(position)

    def distance_to_point(self, x, y, z=None):
        if self.z is None:
            return np.linalg.norm(np.array([x, y]) - np.array(self.xy))
        else:
            return np.linalg.norm(np.array([x, y, z]) - np.array(self.
                    xyz))

    def azimuth_to_point(self, x, y):
        return np.degrees(np.arctan2(y - self.y, x - self.x))

    def inclination_to_point(self, x, y, z):
        try:
            return np.degrees(np.arcsin((z - self.z) / self.
                    distance_to_point(x, y, z)))
        except ZeroDivisionError:
            return 0

    def distance_visibility(self, voxel):
        return 1

    def angle_visibility(self, voxel):
```

```
        return 1

    def visibility(self, voxel):
        return self.distance_visibility(voxel) * self.angle_visibility(
               voxel)
```

Izvorišni kod A.7: Klasa PointSensor

```
from SensorPositioning.sensors.basesensor import BaseSensor


class PointSensor(BaseSensor):
    def __init__(self, initial_pose):
        if isinstance(initial_pose, list):
            pose = initial_pose
        elif initial_pose == 2:
            pose = [0, 0]
        elif initial_pose == 3:
            pose = [0, 0, 0]

        super().__init__(pose)

        self.pose = pose

    @property
    def pose(self):
        if self.z is not None:
            return list(self.xyz)
        return list(self.xy)

    @pose.setter
    def pose(self, value):
        if len(value) == 2:
            self.xy = value[:2]
        else:
            self.xyz = value[:3]
```

Izvorišni kod A.8: Klasa DirectionalSensor

```
import numpy as np

from SensorPositioning.sensors.basesensor import BaseSensor


class DirectionalSensor(BaseSensor):
    def __init__(self, initial_pose):
        if isinstance(initial_pose, list):
            pose = initial_pose
        elif initial_pose == 2:
            pose = [0, 0, 0]
        elif initial_pose == 3:
            pose = [0, 0, 0, 0, 0]

        if len(pose) == 3:
            super().__init__(pose[:2])
        else:
            super().__init__(pose[:3])

        self.pose = pose

    def get_rotated_voxel(self, voxel):
        if self.z is not None:
            position = np.array(voxel.xyz, dtype=np.float64) - np.array(
                    self.xyz)
        else:
            position = np.array(voxel.xy + [0], dtype=np.float64) - np.
                    array(self.xy + [0])

        if self.pitch is not None:
            P = np.radians(self.pitch)
```

```
            position = position * np.matrix([
                [ np.cos(P), 0, np.sin(P)],
                [         0, 1,         0],
                [-np.sin(P), 0, np.cos(P)]
            ])

        if self.yaw is not None:
            Y = np.radians(self.yaw)
            position = position * np.matrix([
                [np.cos(Y), -np.sin(Y), 0],
                [np.sin(Y),  np.cos(Y), 0],
                [         0,          0, 1]
            ])

        if self.z is not None:
            position += np.array(self.xyz)
            position = list(position.flat)
            return [position[0], position[1], position[2]]
        else:
            position += np.array(self.xy + [0])
            position = list(position.flat)
            return [position[0], position[1]]

    def azimuth_visibility(self, x, y, z=None):
        return 1

    def inclination_visibility(self, x, y, z=None):
        return 1

    def angle_visibility(self, voxel):
        position = self.get_rotated_voxel(voxel)
        return self.azimuth_visibility(*position) * self.
               inclination_visibility(*position)

    @property
    def pose(self):
        if self.z is not None:
            return list(self.xyz) + [self.__pitch, self.__yaw]
        return list(self.xy) + [self.__yaw]

    @property
    def pitch(self):
        return self.__pitch

    @property
    def yaw(self):
        return self.__yaw

    @pose.setter
    def pose(self, value):
        if len(value) == 3:
            self.xy = value[:2]
            self.__pitch = None
            self.__yaw = value[2]
        else:
            self.xyz = value[:3]
            self.__pitch = value[3]
            self.__yaw = value[4]
```

Izvorišni kod A.9: Klasa BluetoothLowEnergy

```
import numpy as np
np.seterr(divide='raise')

from SensorPositioning.sensors.pointsensor import PointSensor


class BluetoothLowEnergy(PointSensor):
    def __init__(self, initial_pose, **params):
        super().__init__(initial_pose)
```

```
        self.__rssi1m = params["rssi1m"]
        self.__noise = params["noise"]

        self.__maxrssi = -20 * np.log10(0.1) + self.__rssi1m

    def __rssi(self, voxel):
        try:
            return -20 * np.log10(self.distance_to_point(*voxel.xyz)) +
                   self.__rssi1m
        except FloatingPointError:
            return 1

    def __snr(self, voxel):
        return self.__rssi(voxel) - self.__noise

    def __snr_capped(self, voxel):
        snr = self.__snr(voxel) / (self.__maxrssi - self.__noise)

        if snr > 1:
            return 1
        elif snr < 0:
            return 0

        return snr

    def distance_visibility(self, voxel):
        return self.__snr_capped(voxel)
```

Izvorišni kod A.10: Klasa StereoCamera

```
import numpy as np
np.seterr(divide='raise')

from SensorPositioning.sensors.directionalsensor import
       DirectionalSensor


class StereoCamera(DirectionalSensor):
    def __init__(self, initial_pose, **params):
        super().__init__(initial_pose)

        self.__b = params["b"]
        self.__fx = params["fx"]
        self.__x_dim = params["x_dim"]
        self.__y_dim = params["y_dim"]
        self.__nd = params["nd"]
        self.__cx = params["cx"]
        self.__cy = params["cy"]
        self.__threshold = params["threshold"]

        self.__azimuth = self.__init_azimuth_angles()
        self.__inclination = self.__init_inclination_angles()

    def __init_azimuth_angles(self):
        Z = -1 * self.__b * self.__fx
        X1 = -1 * self.__b * (0 - self.__cx)
        X2 = -1 * self.__b * (128 - self.__cx)
        X3 = -1 * self.__b * (self.__x_dim - 128 - self.__cx)
        X4 = -1 * self.__b * (self.__x_dim - self.__cx)

        angle1 = np.degrees(np.arctan2(X1, Z)) - 180
        angle2 = np.degrees(np.arctan2(X2, Z)) - 180
        angle3 = np.degrees(np.arctan2(X3, Z)) + 180
        angle4 = np.degrees(np.arctan2(X4, Z)) + 180

        return angle1, angle2, angle3, angle4

    def __init_inclination_angles(self):
        Z = -1 * self.__b * self.__fx
        Y1 = -1 * self.__b * (0 - self.__cy)
        Y2 = -1 * self.__b * (self.__y_dim - self.__cy)
```

```
        angle1 = np.degrees(np.arctan2(Y1, Z)) - 180
        angle2 = np.degrees(np.arctan2(Y2, Z)) + 180

        return angle1, angle2

    def __disparity_depth_conversion(self, value):
        try:
            return (self.__b * self.__fx) / value
        except FloatingPointError:
            return np.inf

    def __depth_error(self, disparity):
        return self.__disparity_depth_conversion(disparity) - self.
                __disparity_depth_conversion(disparity + 1)

    def distance_visibility(self, voxel):
        try:
            disparity = np.floor(self.__disparity_depth_conversion(self.
                    distance_to_point(*voxel.xyz)))
        except OverflowError:
            disparity = np.inf

        if disparity <= self.__nd:
            error = self.__depth_error(disparity)
        else:
            error = np.inf

        if error <= self.__threshold:
            return 1
        else:
            return self.__threshold / error

    def azimuth_visibility(self, x, y, z=None):
        if self.distance_to_point(x, y, z) == 0:
            return 1

        angle = self.azimuth_to_point(x, y)

        if self.__azimuth[0] <= angle < self.__azimuth[1]:
            return (angle - self.__azimuth[0]) / (self.__azimuth[1] -
                    self.__azimuth[0])
        elif self.__azimuth[1] <= angle < self.__azimuth[2]:
            return 1
        elif self.__azimuth[2] <= angle < self.__azimuth[3]:
            return (self.__azimuth[3] - angle) / (self.__azimuth[3] -
                    self.__azimuth[2])
        else:
            return 0

    def inclination_visibility(self, x, y, z=None):
        if self.pitch is None or self.distance_to_point(x, y, z) == 0:
            return 1

        angle = self.inclination_to_point(x, y, z)

        if self.__inclination[0] <= angle < self.__inclination[1]:
            return 1
        else:
            return 0
```

Izvorišni kod A.11: Klasa Sensor

```
from SensorPositioning.sensors.ble import BluetoothLowEnergy
from SensorPositioning.sensors.stereocamera import StereoCamera


class Sensor():
    def __new__(cls, sensor_name, *args, **kwargs):
        return globals()[sensor_name](*args, **kwargs)
```

Izvorišni kod A.12: Klasa OptimizationTask

```
from NiaPy.task.task import StoppingTask
from NiaPy.task.task import OptimizationType
from SensorPositioning.optimizers.algorithms.algorithm import Algorithm
from SensorPositioning.optimizers.objectives.objective import Objective


class OptimizationTask():
    def __init__(self, objective, evals_per_dimension, algorithm,
        testcase):
        objective = Objective(objective["name"], testcase)

        dimensionality = testcase.get_optimization_dimensionality()
        evaluations = evals_per_dimension * dimensionality

        self.__task = StoppingTask(D=dimensionality, nFES=evaluations,
            optType=OptimizationType.MINIMIZATION, benchmark=
            objective)

        self.__algorithm = Algorithm(algorithm['name'], **algorithm['
            params'])

    def run(self):
        return self.__algorithm.run(task=self.__task)
```

Izvorišni kod A.13: Klasa Algorithm

```
from NiaPy.algorithms.basic import ArtificialBeeColonyAlgorithm
from NiaPy.algorithms.basic import FireworksAlgorithm
from NiaPy.algorithms.basic import ParticleSwarmOptimization
from NiaPy.algorithms.other import HillClimbAlgorithm
from NiaPy.algorithms.other import NelderMeadMethod
from NiaPy.algorithms.other import SimulatedAnnealing

class Algorithm():
    def __new__(cls, algorithm_name, *args, **kwargs):
        return globals()[algorithm_name](*args, **kwargs)
```

Izvorišni kod A.14: Klasa Objective

```
from NiaPy.benchmarks.benchmark import Benchmark

from SensorPositioning.optimizers.objectives.area_coverage import
    AreaCoverage


class Objective(Benchmark):
    def __init__(self, objective, testcase):
        self.__objective = globals()[objective](testcase)

        self.Lower, self.Upper = testcase.get_optimization_bounds()

    def function(self):
        def evaluate(D, sol):
            return self.__objective.penalty(sol)
        return evaluate
```

Izvorišni kod A.15: Klasa AreaCoverage

```
class AreaCoverage():
    def __init__(self, testcase):
        self.__testcase = testcase

    def penalty(self, sol):
        self.__testcase.set_sensor_positions(sol)

        sum = 0
```

```
        for v in self.__testcase.voxels:
            v.loss = 1
            for s in self.__testcase.sensors:
                if self.__testcase.environment.line_of_sight(s, v):
                    v.loss *= (1 - s.visibility(v))

            sum += v.loss

        sum /= len(self.__testcase.voxels)

        self.__testcase.add_step_result(sol, sum)

        return sum
```

Izvorišni kod A.16: Klasa Testcase

```
from SensorPositioning.geometry.mesh import Mesh
from SensorPositioning.geometry.environment import Environment
from SensorPositioning.sensors.sensor import Sensor
from SensorPositioning.sensors.directionalsensor import
       DirectionalSensor


class Testcase():
    def __init__(self, space):
        self.__space = space

        self.__environment = None
        self.__sensors = {}

        self.__results = []

    def __get_mesh(self, data):
        polygon = data["polygon"]

        if self.__space == 2:
            return Mesh(polygon)
        else:
            return Mesh(polygon, z_coords=(data["bottom"], data["top"]))

    def add_environment(self, environment, voxelsize):
        self.__environment = Environment(voxelsize)

        for data in environment['positivemeshes']:
            self.__environment.add_positive_mesh(self.__get_mesh(data))

        for data in environment['negativemeshes']:
            if "polygon" in data.keys() and data["polygon"] != []:
                self.__environment.add_negative_mesh(self.__get_mesh(
                        data))

        self.__environment.create_environment()

    def __add_sensor(self, sensor):
        for i in range(len(self.__sensors.keys()) + 1):
            sid = "sensor" + str(i)
            if sid not in self.__sensors.keys():
                break

        self.__sensors[sid] = sensor

    def add_sensors(self, sensors):
        for s in sensors:
            self.__add_sensor(Sensor(s["name"], self.__space, **s["
                    params"]))

    def get_optimization_bounds(self):
        lower_bounds = []
        upper_bounds = []

        for s in self.__sensors.values():
            lower_bounds += list(self.__environment.lower_bounds)
```

```
            upper_bounds += list(self.__environment.upper_bounds)

            if isinstance(s, DirectionalSensor):
                if self.__space == 3:
                    lower_bounds += [-90]
                    upper_bounds += [90]

                lower_bounds += [-180]
                upper_bounds += [180]

        return lower_bounds, upper_bounds

    def get_optimization_dimensionality(self):
        lower_bounds, _ = self.get_optimization_bounds()

        return len(lower_bounds)

    def set_sensor_positions(self, pose_vector):
        i = 0
        for s in self.__sensors.values():
            j = self.__space
            if isinstance(s, DirectionalSensor):
                j += 1
                if self.__space == 3:
                    j += 1

            s.pose = pose_vector[i:i+j]
            i += j

    def add_step_result(self, solution, score):
        self.__results.append((solution, score))

    @property
    def voxels(self):
        return self.__environment.voxels

    @property
    def environment(self):
        return self.__environment

    @property
    def sensors(self):
        return self.__sensors.values()

    @property
    def results(self):
        return self.__results

    @property
    def sensor_poses(self):
        pose_vector = []
        for s in self.__sensors.values():
            pose_vector += s.pose
        return pose_vector
```

Izvorišni kod A.17: Klasa Config

```
import json


class Config():
    def __init__(self, file_path):
        f = open(file_path, "r")

        self.__datastorage = json.load(f)

    def get_section(self, section):
        if section in self.__datastorage.keys():
            return self.__datastorage[section]
        return None
```

Izvorišni kod A.18: Klasa Processing

```
import os
import timeit
import pickle

from SensorPositioning.processing.testcase import Testcase
from SensorPositioning.optimizers.optimization import OptimizationTask


class Processing():
    def __init__(self, config):
        self.__environment_definitions = config.get_section("
                environments")
        self.__algorithm_definitions = config.get_section("algorithms")
        self.__sensor_definitions = config.get_section("sensors")
        self.__objective_definitions = config.get_section("objectives")

        processing = config.get_section("processing")
        self.__name = processing["name"]
        self.__save_path = processing["save_path"]
        self.__display = processing["display"]
        self.__threadnumber = processing["threadnumber"]

        self.__environments = processing["environments"]
        self.__spaces = processing["spaces"]
        self.__sensors = self.__get_sensors_list(processing["sensors"])
        self.__algorithms = processing["algorithms"]
        self.__evals_per_dimension = processing["evals_per_dimension"]
        self.__objectives = processing["objectives"]
        self.__voxelsizes = processing["voxelsizes"]
        self.__iterations = processing["iterations"]

        self.__optimization_tasks = []

    def __get_sensors_list(self, sensors):
        sensors_list = []

        for s in sensors:
            for v in s["numbers"]:
                sensors_list.append([self.__sensor_definitions[s["type"]
                    ] for _ in range(v)])

        return sensors_list

    def generate_testcases(self):
        os.makedirs(self.__save_path, exist_ok=True)
        os.makedirs(f'{self.__save_path}/started/', exist_ok=True)
        os.makedirs(f'{self.__save_path}/done/', exist_ok=True)
        os.makedirs(f'{self.__save_path}/errors/', exist_ok=True)

        for iteration in range(self.__iterations):
            for space in self.__spaces:
                for environment in self.__environments:
                    for sensors in self.__sensors:
                        for algorithm in self.__algorithms:
                            for evals_per_dimension in self.
                                    __evals_per_dimension:
                                for objective in self.__objectives:
                                    for voxelsize in self.__voxelsizes:
                                        optimization_params = {
                                            'save_path': self.
                                                    __save_path,
                                            'space': space,
                                            'iteration': iteration,
                                            'environment': self.
                                                    __environment_definitions
                                                    [environment],
                                            'voxelsize': voxelsize,
                                            'sensors': sensors,
                                            'objective': self.
                                                    __objective_definitions
                                                    [objective],
```

```
                                        'evals_per_dimension':
                                                evals_per_dimension,
                                        'algorithm': self.
                                                __algorithm_definitions
                                                [algorithm]
                                    }

                                    opt_data = Processing.
                                            get_opt_data(
                                            optimization_params)
                                    if not os.path.isfile(f"{
                                            optimization_params['
                                            save_path']}/started/{
                                            opt_data}.pickle"):
                                        self.__optimization_tasks.
                                                append(
                                                optimization_params)

    @staticmethod
    def get_opt_data(op):
        opt_data = f"{op['space']}_{op['environment']['name']}_{op['
                voxelsize']}_{op['evals_per_dimension']}_{op['objective
                ']['name']}_{op['algorithm']['name']}"

        s = {}
        for el in op['sensors']:
            if el['name'] not in s.keys():
                s[el['name']] = 1
            else:
                s[el['name']] += 1

        for k, v in s.items():
            opt_data = f"{opt_data}_{k}_{v}"

        opt_data = f"{opt_data}_{op['iteration']:04d}"

        return opt_data

    @staticmethod
    def runner(op):
        testcase = Testcase(op['space'])
        testcase.add_environment(op['environment']['definition'], op['
                voxelsize'])
        testcase.add_sensors(op['sensors'])

        ot = OptimizationTask(op['objective'], op['evals_per_dimension']
                , op['algorithm'], testcase)

        opt_data = Processing.get_opt_data(op)

        if not os.path.isfile(f"{op['save_path']}/started/{opt_data}.
                pickle"):
            with open(f"{op['save_path']}/started/{opt_data}.pickle", '
                    bw') as f:
                pickle.dump({}, f)

            results = {
                'optimization_params': op,
            }

            try:
                results['times'] = {}
                results['scores'] = {}

                results['times']['start_time'] = timeit.default_timer()
                results['scores']['final_score'] = ot.run()
                results['times']['end_time'] = timeit.default_timer()
                results['scores']['step_results'] = testcase.results

                with open(f"{op['save_path']}/done/{opt_data}.pickle", '
                        bw') as f:
                    pickle.dump(results, f)
```

```
            except:
                with open(f"{op['save_path']}/errors/{opt_data}.pickle",
                          'bw') as f:
                    pickle.dump(results, f)

    def run(self):
        self.generate_testcases()

        try:
            threads_num = int(os.environ['SLURM_NNODES']) * int(os.
                  environ['SLURM_NTASKS_PER_NODE'])
        except:
            threads_num = self.__threadnumber

        try:
            from mpi4py.futures import MPIPoolExecutor

            p = MPIPoolExecutor(threads_num - 1)
        except ImportError:
            from multiprocessing import Pool

            p = Pool(threads_num)

        p.map(Processing.runner, self.__optimization_tasks)

    @property
    def generated_tasks(self):
        return len(self.__optimization_tasks)
```

B. Tablice s rezultatima

Tablica B.1: Srednje vrijednosti pokrivenosti prostora C za Ispitni prostor 1

Rast.	Osjetila		Pokrivenost prostora C					
[m]	Tip	Broj	ABC	FWA	PSO	HCA	NMM	SA
2D								
0,1	Estimote BLE	1	0,1440	0,1430	0,1445	0,1442	0,1271	0,1351
		2	0,2715	0,2570	0,2723	0,2685	0,2399	0,2513
		3	0,3660	0,3457	0,3664	0,3613	0,3323	0,3458
	Stereolabs ZED	1	0,4284	0,3676	0,3973	0,4362	0,2564	0,2647
		2	0,6763	0,5672	0,5582	0,6607	0,4197	0,4670
		3	0,8147	0,6970	0,6776	0,7896	0,5596	0,5955
0,5	Estimote BLE	1	0,1299	0,1288	0,1302	0,1298	0,1167	0,1231
		2	0,2454	0,2334	0,2460	0,2423	0,2227	0,2269
		3	0,3345	0,3173	0,3350	0,3288	0,3039	0,3175
	Stereolabs ZED	1	0,3827	0,3274	0,3470	0,3912	0,2424	0,2369
		2	0,6122	0,5243	0,5207	0,6065	0,4023	0,4326
		3	0,7738	0,6618	0,6455	0,7432	0,5361	0,5607
1,0	Estimote BLE	1	0,1193	0,1153	0,1195	0,1182	0,1020	0,1115
		2	0,2283	0,2096	0,2278	0,2200	0,1983	0,2074
		3	0,3178	0,2900	0,3176	0,3013	0,2777	0,2893
	Stereolabs ZED	1	0,3414	0,3044	0,3100	0,3466	0,2314	0,2412
		2	0,5719	0,4969	0,4882	0,5594	0,3955	0,4236
		3	0,7385	0,6238	0,6009	0,7057	0,5226	0,5508
3D								
0,1	Estimote BLE	1	0,0792	0,0765	0,0795	0,0790	0,0671	0,0694
		2	0,1510	0,1369	0,1512	0,1462	0,1270	0,1321
		3	0,2085	0,1872	0,2087	0,2004	0,1719	0,1822
	Stereolabs ZED	1	0,2741	0,2262	0,2247	0,2739	0,1558	0,1516
		2	0,4715	0,3645	0,3243	0,4510	0,2586	0,2786
		3	0,5917	0,4580	0,4024	0,5538	0,3481	0,3670
0,5	Estimote BLE	1	0,0669	0,0651	0,0670	0,0667	0,0592	0,0612
		2	0,1283	0,1169	0,1283	0,1244	0,1087	0,1145
		3	0,1780	0,1621	0,1779	0,1715	0,1509	0,1592
	Stereolabs ZED	1	0,2389	0,2004	0,1951	0,2408	0,1371	0,1461
		2	0,4220	0,3273	0,2939	0,3970	0,2372	0,2579
		3	0,5407	0,4071	0,3767	0,5001	0,3173	0,3326
1,0	Estimote BLE	1	0,0541	0,0534	0,0543	0,0541	0,0497	0,0503
		2	0,1055	0,0977	0,1056	0,1025	0,0929	0,0953
		3	0,1489	0,1364	0,1491	0,1434	0,1314	0,1359
	Stereolabs ZED	1	0,2078	0,1749	0,1712	0,2062	0,1262	0,1261
		2	0,3696	0,2868	0,2629	0,3516	0,2169	0,2289
		3	0,4817	0,3680	0,3357	0,4444	0,2937	0,3151

Tablica B.2: Srednje vrijednosti pokrivenosti prostora C za Ispitni prostor 2

Rast.	Osjetila		Pokrivenost prostora C					
[m]	Tip	Broj	ABC	FWA	PSO	HCA	NMM	SA
2D								
0,1	Estimote BLE	1	0,1188	0,1164	0,1188	0,1183	0,0968	0,1082
		2	0,2069	0,2008	0,2070	0,2049	0,1918	0,2007
		3	0,3015	0,2809	0,2960	0,2909	0,2640	0,2822
	Stereolabs ZED	1	0,2924	0,2216	0,2337	0,2691	0,1715	0,1807
		2	0,5241	0,3682	0,3689	0,4476	0,2862	0,3203
		3	0,6805	0,4889	0,4824	0,5817	0,3816	0,4235
0,5	Estimote BLE	1	0,0973	0,0962	0,0974	0,0970	0,0883	0,0956
		2	0,1927	0,1864	0,1920	0,1889	0,1737	0,1842
		3	0,2848	0,2705	0,2807	0,2766	0,2490	0,2671
	Stereolabs ZED	1	0,2596	0,2016	0,2184	0,2292	0,1529	0,1790
		2	0,4502	0,3631	0,3599	0,4105	0,2758	0,3221
		3	0,6093	0,4858	0,4611	0,5485	0,3701	0,4378
1,0	Estimote BLE	1	0,1082	0,1072	0,1080	0,1081	0,0846	0,1044
		2	0,2158	0,2050	0,2135	0,2118	0,1667	0,1958
		3	0,3206	0,2968	0,3139	0,3085	0,2342	0,2702
	Stereolabs ZED	1	0,2143	0,2113	0,2118	0,2151	0,1654	0,1936
		2	0,4199	0,3987	0,3757	0,4105	0,3045	0,3543
		3	0,6108	0,5271	0,5005	0,5803	0,4140	0,4769
3D								
0,1	Estimote BLE	1	0,0812	0,0794	0,0814	0,0809	0,0709	0,0713
		2	0,1526	0,1432	0,1533	0,1484	0,1386	0,1300
		3	0,2101	0,1971	0,2105	0,2016	0,1958	0,1771
	Stereolabs ZED	1	0,2651	0,2278	0,2228	0,2736	0,1608	0,1572
		2	0,4387	0,3406	0,3233	0,4308	0,2516	0,2635
		3	0,5555	0,4200	0,3856	0,5237	0,3272	0,3467
0,5	Estimote BLE	1	0,0682	0,0666	0,0683	0,0679	0,0606	0,0623
		2	0,1328	0,1200	0,1325	0,1292	0,1159	0,1146
		3	0,1836	0,1647	0,1839	0,1762	0,1626	0,1580
	Stereolabs ZED	1	0,2339	0,2023	0,2030	0,2412	0,1470	0,1433
		2	0,4000	0,3112	0,2962	0,3825	0,2396	0,2501
		3	0,5159	0,3925	0,3668	0,4771	0,3141	0,3389
1,0	Estimote BLE	1	0,0597	0,0565	0,0600	0,0588	0,0472	0,0514
		2	0,1176	0,1015	0,1149	0,1127	0,0921	0,0966
		3	0,1616	0,1400	0,1613	0,1539	0,1276	0,1368
	Stereolabs ZED	1	0,2117	0,1754	0,1711	0,2089	0,1304	0,1320
		2	0,3671	0,2843	0,2651	0,3444	0,2198	0,2327
		3	0,4723	0,3592	0,3387	0,4263	0,2872	0,3112

Tablica B.3: Srednje vrijednosti pokrivenosti prostora C za Ispitni prostor 3

Rast. [m]	Osjetila Tip	Broj	Pokrivenost prostora C ABC	FWA	PSO	HCA	NMM	SA
2D								
0,1	Estimote BLE	1	0,1350	0,1346	0,1357	0,1353	0,1150	0,1244
		2	0,2590	0,2413	0,2586	0,2563	0,2268	0,2328
		3	0,3472	0,3228	0,3433	0,3365	0,3038	0,3220
	Stereolabs ZED	1	0,3779	0,3734	0,3454	0,3953	0,2213	0,2244
		2	0,5951	0,5237	0,4900	0,5887	0,3849	0,4062
		3	0,7645	0,6122	0,5934	0,7079	0,4886	0,5254
0,5	Estimote BLE	1	0,1257	0,1235	0,1262	0,1253	0,1037	0,1174
		2	0,2438	0,2263	0,2457	0,2410	0,2064	0,2196
		3	0,3342	0,3068	0,3289	0,3225	0,2840	0,3027
	Stereolabs ZED	1	0,3680	0,3533	0,3193	0,3737	0,2086	0,2313
		2	0,5812	0,5059	0,4799	0,5601	0,3732	0,4004
		3	0,7516	0,6034	0,5839	0,6799	0,4748	0,5132
1,0	Estimote BLE	1	0,1211	0,1156	0,1215	0,1207	0,0960	0,1104
		2	0,2357	0,2130	0,2319	0,2274	0,1869	0,2057
		3	0,3371	0,3015	0,3250	0,3208	0,2621	0,2905
	Stereolabs ZED	1	0,3391	0,3382	0,2971	0,3550	0,2057	0,2300
		2	0,5614	0,4949	0,4710	0,5495	0,3760	0,4047
		3	0,7557	0,6036	0,5764	0,6733	0,4721	0,5276
3D								
0,1	Estimote BLE	1	0,0737	0,0723	0,0740	0,0735	0,0635	0,0670
		2	0,1422	0,1297	0,1434	0,1370	0,1238	0,1204
		3	0,1952	0,1775	0,1956	0,1854	0,1731	0,1668
	Stereolabs ZED	1	0,2475	0,2140	0,2001	0,2442	0,1311	0,1468
		2	0,3978	0,3217	0,2814	0,3824	0,2224	0,2412
		3	0,5276	0,3848	0,3424	0,4634	0,2866	0,3159
0,5	Estimote BLE	1	0,0638	0,0615	0,0640	0,0633	0,0548	0,0577
		2	0,1243	0,1097	0,1229	0,1205	0,1049	0,1059
		3	0,1721	0,1506	0,1719	0,1639	0,1462	0,1497
	Stereolabs ZED	1	0,2235	0,1985	0,1821	0,2237	0,1231	0,1165
		2	0,3712	0,3027	0,2592	0,3491	0,2142	0,2209
		3	0,4929	0,3627	0,3236	0,4285	0,2690	0,2945
1,0	Estimote BLE	1	0,0528	0,0507	0,0533	0,0523	0,0439	0,0476
		2	0,1032	0,0927	0,1021	0,1014	0,0837	0,0910
		3	0,1482	0,1298	0,1460	0,1414	0,1204	0,1299
	Stereolabs ZED	1	0,1942	0,1769	0,1536	0,1957	0,1142	0,1168
		2	0,3333	0,2737	0,2383	0,3114	0,1936	0,2115
		3	0,4529	0,3481	0,3011	0,3842	0,2512	0,2735

Tablica B.4: Srednje vrijednosti vremena izvršavanja za Ispitni prostor 1

Rast. [m]	Osjetila Tip	Broj	Vrijeme [s] ABC	FWA	PSO	HCA	NMM	SA
2D								
0,1	Estimote BLE	1	210	211	200	206	212	203
		2	855	867	834	849	847	834
		3	1949	1976	1959	1921	1943	1894
	Stereolabs ZED	1	401	358	381	380	383	363
		2	1614	1460	1536	1536	1491	1472
		3	3625	3292	3448	3424	3247	3319
0,5	Estimote BLE	1	11	11	10	10	11	11
		2	44	44	43	43	44	42
		3	101	99	100	98	99	95
	Stereolabs ZED	1	20	18	19	19	19	18
		2	81	74	76	77	77	74
		3	182	167	172	173	168	166
1,0	Estimote BLE	1	3	3	3	3	3	3
		2	14	14	14	14	14	14
		3	32	32	32	31	31	30
	Stereolabs ZED	1	6	6	6	6	6	6
		2	25	24	24	24	25	24
		3	57	53	54	55	54	53
3D								
0,1	Estimote BLE	1	18431	18461	17995	18098	18720	17838
		2	73940	75445	73961	74147	76056	72374
		3	168120	169525	166848	167207	166914	164474
	Stereolabs ZED	1	41398	36574	39299	39556	39533	37184
		2	166748	148269	156533	158853	150577	149291
		3	377526	332159	355918	354330	327424	337660
0,5	Estimote BLE	1	209	210	203	204	210	200
		2	842	846	835	832	850	813
		3	1894	1911	1902	1875	1878	1830
	Stereolabs ZED	1	461	410	433	441	441	410
		2	1866	1661	1770	1771	1710	1666
		3	4198	3765	3945	3949	3821	3739
1,0	Estimote BLE	1	37	37	36	36	37	36
		2	147	150	147	145	149	142
		3	331	337	339	327	331	322
	Stereolabs ZED	1	79	71	76	77	77	71
		2	321	290	307	308	302	288
		3	719	655	690	686	669	653

Tablica B.5: Srednje vrijednosti vremena izvršavanja za Ispitni prostor 2

Rast. [m]	Osjetila Tip	Broj	Vrijeme [s] ABC	FWA	PSO	HCA	NMM	SA
2D								
0,1	Estimote BLE	1	398	386	396	378	403	382
		2	1581	1548	1583	1519	1582	1516
		3	3609	3463	3565	3444	3605	3421
	Stereolabs ZED	1	631	591	612	613	642	595
		2	2539	2396	2484	2445	2517	2367
		3	5749	5328	5658	5499	5539	5368
0,5	Estimote BLE	1	16	16	16	16	16	15
		2	64	62	64	62	64	61
		3	145	139	145	140	143	137
	Stereolabs ZED	1	26	25	25	25	26	24
		2	103	96	101	99	101	96
		3	233	217	228	220	220	216
1,0	Estimote BLE	1	4	4	4	4	4	4
		2	15	15	15	15	15	14
		3	34	33	34	33	33	33
	Stereolabs ZED	1	6	6	6	6	6	6
		2	24	23	24	23	24	23
		3	55	51	55	53	52	51
3D								
0,1	Estimote BLE	1	44691	42508	42978	43273	45124	41111
		2	184370	169933	202790	173249	177205	166255
		3	419565	383022	441177	388977	391744	373722
	Stereolabs ZED	1	80246	78205	81395	81430	82355	76552
		2	324296	312148	321681	321507	309404	306949
		3	732363	704010	722318	714865	675668	690885
0,5	Estimote BLE	1	444	400	469	416	432	389
		2	1801	1596	1932	1646	1719	1568
		3	4096	3547	4355	3672	3807	3513
	Stereolabs ZED	1	745	733	766	758	745	720
		2	3036	2937	3069	2998	3010	2882
		3	6841	6562	6828	6699	6537	6477
1,0	Estimote BLE	1	63	56	64	60	61	55
		2	253	228	266	235	240	221
		3	573	501	595	525	516	497
	Stereolabs ZED	1	106	103	106	105	107	101
		2	427	415	430	420	415	409
		3	964	930	958	941	938	914

Tablica B.6: Srednje vrijednosti vremena izvršavanja za Ispitni prostor 3

Rast. [m]	Osjetila Tip	Broj	Vrijeme [s] ABC	FWA	PSO	HCA	NMM	SA
2D								
0,1	Estimote BLE	1	341	326	346	327	343	324
		2	1370	1314	1392	1305	1393	1286
		3	3133	2983	3079	2948	3110	2889
	Stereolabs ZED	1	543	529	525	516	560	510
		2	2239	2121	2154	2082	2215	2069
		3	5039	4768	4881	4682	4726	4658
0,5	Estimote BLE	1	14	14	14	14	14	14
		2	57	56	55	55	58	54
		3	131	124	127	125	131	122
	Stereolabs ZED	1	23	22	22	22	24	22
		2	95	89	92	88	93	87
		3	214	201	208	197	207	197
1,0	Estimote BLE	1	4	4	4	4	4	4
		2	15	15	15	15	15	14
		3	34	33	34	33	33	32
	Stereolabs ZED	1	6	6	6	6	6	6
		2	25	24	24	23	25	23
		3	56	53	55	52	56	51
3D								
0,1	Estimote BLE	1	29290	27345	29405	27398	29285	26567
		2	117252	109857	118468	109568	119281	107872
		3	264958	248938	265473	246061	268506	243814
	Stereolabs ZED	1	52724	50288	51385	49920	54005	49007
		2	216428	203601	210044	199864	209281	196872
		3	488887	458246	475226	451613	470601	447078
0,5	Estimote BLE	1	265	256	267	255	273	247
		2	1063	1026	1034	1021	1092	1004
		3	2404	2315	2282	2308	2440	2275
	Stereolabs ZED	1	490	469	473	462	506	458
		2	2015	1913	1941	1864	2015	1833
		3	4579	4288	4369	4194	4422	4161
1,0	Estimote BLE	1	38	37	37	37	38	37
		2	152	148	149	148	151	145
		3	341	335	333	334	323	325
	Stereolabs ZED	1	71	67	69	66	73	66
		2	287	276	278	267	290	262
		3	653	617	631	603	643	596

ŽIVOTOPIS

Diego Sušanj rođen je 1.11.1991. godine u Rijeci. Na Tehničkom fakultetu Sveučilišta u Rijeci upisuje preddiplomski sveučilišni studij računarstva 2010. te završava 2013. godine. Svoje obrazovanje nastavlja na istom fakultetu kroz diplomski sveučilišni studij računarstva, modul računalni sustavi.

Po završetku diplomskog studija računarstva 2015. godine zapošljava se na Tehničkom fakultetu Sveučilišta u Rijeci na radnom mjestu asistent pri Zavodu za računarstvo. Iste godine upisuje poslijediplomski doktorski studije elektrotehnike te se prebacuje na poslijediplomski studij računarstva po njegovom otvaranju 2018. godine. Uz nastavne i znanstvene obaveze, sudjelovao je i na projektima Provjere inovativnog koncepta 6 te Sveučilišne potpore pod vodstvom svog mentora izv. prof. Kristijana Lenca.

POPIS RADOVA

[1] D. Sušanj, D. Pinčić, i K. Lenac, "Effective Area Coverage of 2D and 3D Environments With Directional and Isotropic Sensors", *IEEE Access*, svezak 8, str. 185 595–185 608, 2020.

[2] A. Ramakić, D. Sušanj, K. Lenac, i Z. Bundalo, "Depth-Based Real-Time Gait Recognition", *Journal of Circuits, Systems and Computers*, str. 2050266, 2020.

[3] K. Lenac, D. Sušanj, A. Ramakić, i D. Pinčić, "Extending Appearance Based Gait Recognition with Depth Data", *Applied Sciences*, svezak 9, broj 24, str. 5529, 2019.

[4] A. Hrovatič, K. Kwon, D. Sušanj, P. Peer, i Ž. Emeršič, "Generation of 2D ear dataset with annotated view angles as a basis for angle-aware ear recognition", u *Proceedings of the 28th International Electrotechnical and Computer Science Conference (2019)*, 2019, str. 4.

[5] F. Hržić, D. Sušanj, i K. Lenac, "Optimal beacon positioning for indoor drone navigation", u *12th Annual Baška GNSS Conference (2018)*, 2018, str. 109–117.

[6] F. Hržić, V. Jansky, D. Sušanj, G. Gulan, I. Kožar, i Ž. Jeričević, "Information entropy measures and clustering improve edge detection in medical X-ray images", u *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. IEEE, 2018, str. 0164–0166.

[7] A. Ramakić, D. Sušanj, i K. Lenac, "Geolocation using Google Vision API", u *10th Annual Baška GNSS Conference (2016)*, 2017, str. 55–66.

[8] D. Sušanj i D. Arbula, "Distributed Graph Reduction Algorithm with Parallel Rigidity Maintenance", u *2016 39th International Convention on Information and Com-*

munication Technology, Electronics and Microelectronics (MIPRO). IEEE, 2016, str. 237–240.

[9] D. Sušanj, G. Gulan, I. Kožar, i Ž. Jeričević, "Bone shape characterization using the fourier transform and edge detection in digital X-ray images", u *2016 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO).* IEEE, 2016, str. 362–364.

[10] D. Sušanj, V. Tuhtan, L. Lenac, G. Gulan, I. Kožar, i Ž. Jeričević, "Using entropy information measures for edge detection in digital images", u *2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO).* IEEE, 2015, str. 352–355.