Mobil ágensek navigációjának vizsgálata szimulációs környezetekben

Doktori értekezés

Bírálói vélemények alapján javított változat

Szabó Richárd

Eötvös Loránd Tudományegyetem Informatikai Kar
Programozáselmélet és Szoftvertechnológiai Tanszék

Eötvös Loránd Tudományegyetem Informatika Doktori Iskola
Az informatika alapjai és módszertana program
Iskola- és programvezető: Dr. Demetrovics János akadémikus

Témavezető:
Dr. Kampis György
tanszékvezető egyetemi docens

Budapest, 2006., 2008.

Tartalomjegyzék

Köszönetnyilvánítás

Ez a doktori dolgozat sok év munkájának az eredménye. Ez alatt az idő alatt sok ember segített előre a célom felé, amiért hálás vagyok nekik.

Köszönettel tartozom témavezetőmnek, Kampis Györgynek, aki a kezdetektől alapvető iránymutatást adott, és a komoly döntések meghozatalában támogatott. Köszönöm Gulyás Lászlónak, hogy módszertani tanácsaival segített, és együtt dolgozott velem egy érdekes területen. Köszönöm Istenes Zoltánnak, hogy a Mobil robotok szimulációja című könyvem utómunkálataiban részt vett, és lelkiismeretesen lektorálta azt.

Köszönöm Olivier Michelnek, a Cyberbotics cég vezetőjének a Webots szimulációs környezetben végzett munkámmal kapcsolatos együttgondolkodást és támogatást.

Hálás vagyok Salamon Andrásnak, aki szinte az összes írásomat elolvasta, és hasznos tanácsokkal, megjegyzésekkel javította azokat.

Köszönöm szüleimnek, amiért biztosították nekem a tanulás feltételeit, és elültették bennem a tudás iránti vágyat, és feleségemnek, hogy mellettem állt és biztatott.

Szabó Richárd
2006. augusztus

Bevezetés

Az ember a történelem előtti időktől kezdve mindig jólétet próbál magának biztosítani. Ehhez egészen a legutóbbi időkig mások fizikai és szellemi teljesítményének hasznosításán át vezetett az út. Az elmúlt kétszáz év tudományos-technikai fejlődése merőben új lehetőségeket teremt. Az ipari forradalom és annak huszadik századi kibontakozása létrehozta a „fizikai testet" és a „mozgatóerőt". Az informatikai forradalom annak küszöbén áll, hogy létrehozza a „szellemet", mely a test önálló irányítására lesz képes.

Mivel a létező dolgok folyamatos mozgásban, változásban vannak, akárcsak az észlelő maga, ezért a testet irányító intelligencia megvalósításához elengedhetetlen a navigáció megoldása. Ezáltal a tárgyak, élőlények térbeli elhelyezkedése, viszonyai tisztázhatók, és a cselekvőt magában foglaló környezet megismerhető, mely lényeges a további feladatvégzés szempontjából.

A navigáció mint fogalom jelentése — különösen robotok esetében — nem teljesen egyezik meg a megszokott szóhasználattal. A köznapi jelentés ismert vagy térképpel könnyen felmérhető környezetben végzett optimális haladás megvalósítását takarja. A csillagászati vagy tengeri navigáció ezzel szemben az észlelő és úticélja helyének meghatározását emeli ki, a többi részfeladatot az ember kogníciós folyamataira bízza. A hasonló jelentésű tájékozódás szavunk is ehhez a tartalomhoz áll közelebb. A kifejezés kettősségét a Magyar értelmező kéziszótár is alátámasztja, melyben a főnévi forma a feladat statikus, az igei alak pedig dinamikus jellegét hangsúlyozza. Mesterséges intelligens ágens esetében a navigáció a teljes folyamatot lefedi, az érzékeléstől a helymeghatározáson át a cselekvés megtervezéséig és kivitelezéséig.

A hatékony navigáció megvalósításához több alternatív út vezet. Kézenfekvő lehetőség a mérnöki, konstruktív megközelítés, mely a „semmiből" a probléma alapos vizsgálatával konstruálja meg a megfelelő reprezentációkat és az azokat felhasználó algoritmusokat. Egy másik hozzáállás a már létező, sikeres eljárások

elemzésén, lényeges mozzanatainak felhasználásán keresztül jut el a megoldáshoz. E létező navigációs módszereket az élőlények szolgáltathatják.

Az informatika, miközben az elme egyéb funkciói mellett a környezet modellezését is megvalósíthatja, a kutatás egy új módszerét is kínálja. A számítógépes szimulációk segítségével a tájékozódással kapcsolatos kísérletek virtuális térben folyhatnak. Ez, azon kívül, hogy költség-, idő- és energiatakarékosabb, jóval rugalmasabb megoldás is, mivel a kísérletek bonyolultsága, valósághűsége szabadon alakítható, ami segítheti a vizsgált eljárások robusztusságának tesztelését, az aktuálisan érdektelen részektől való eltávolodást, és a probléma lényeges elemeire való fókuszálást.

Értekezésemben mesterséges és természetes mobil ágensek, azaz robotok és hangyák navigációját vizsgálom. Először áttekintem, hogy milyen kérdések, nehézségek és előnyök jelentkeznek a szimuláció használatával kapcsolatban, majd a navigáció problémakörét vázolom, egyúttal bemutatom az ismertebb mérnöki és természetes megoldásokat. Ezután ismertetem egy robotikai versenyen elért eredményeimet, az ezzel kapcsolatos tapasztalataimat, melyek a navigáció további vizsgálatára ösztönöztek. A negyedik, ötödik és hatodik fejezetben a robotok foglaltsági hálón alapuló térreprezentálását elemzem. Először bemutatok egy működő diszkrét reprezentációs rendszert, mely különféle környezetek bejárását és feltérképezését végzi, majd ezt bővítem egy topológiai eljárással, illetve kamerát használó navigációval. A hetedik fejezet hangyák ételgyűjtése során a környezetben meglévő információtartalom és a hangyakolónia munkavégzésének kölcsönhatását vizsgálja.

1

Intelligens ágensek szimulációja

1.1. A szimuláció szerepe a kutatásban

A tudományos igényű vizsgálódás, kutatás módszereinek egy viszonylag fiatal képviselője a számítógépes szimuláció. A második világháború végén az atombomba létrehozásáért folytatott versenyben az első számítógépek feladata a nukleáris folyamatok modelljeinek felhasználásával szimulációk futtatása volt. A számítógépek rohamos fejlődésével és széleskörű elterjedésével a szimuláció egyre összetettebb modellek kezelésére vált alkalmassá, és mind gyakoribb elemzési módszerré vált.

Természetesen a szimuláció a hagyományos kutatási módszereket nem váltja föl: a természettudományos megfigyelések végzése, hipotézisek felállítása, tételbizonyítás, kísérletek végzése, predikció továbbra is alapvetőek. A szimuláció csupán ezek kiegészítő eszköze lehet olyan esetekben, amikor

- a probléma mélyebb megértését, más nézőpontból való vizsgálatát teszi lehetővé, például a megfigyelő és a megfigyelt közötti interakció biztosításával vagy a kísérleti környezet vizualizációjával, kézzelfoghatóvá tételével,
- a feladat analitikus megoldása, zárt képletbe rendezése nem lehetséges, vagy aránytalanul komoly erőfeszítést igényel,
- a probléma előzetes vizsgálatához, a hipotézisek felállításához nyújt segítséget, a tételek bizonyítása előtt,
- a problémát vizsgáló kutató, más terület szakértőjeként, nincs fölvértezve

kellő matematikai-fizikai ismeretekkel, mégis a jelenségek leírása mellett azok elemzése is céljai közé tartozik,

- el kell dönteni, hogy a probléma modellje egy érvényes leírás-e, vagyis teljesül-e a koherencia, a konzisztencia és a helyesség követelménye,
- meg kell vizsgálni, hogy a modell alkalmazható-e az adott problémára, azaz a bemenetek, kimenetek és a közöttük lévő kapcsolatok a modellezett jelenséggel összhangban vannak-e ([35]).

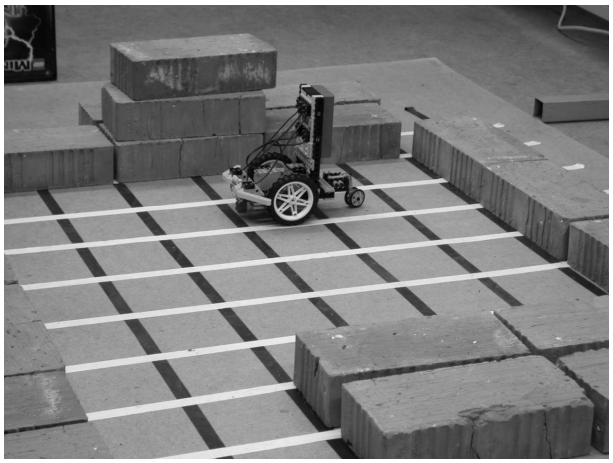
A szimuláció a természettudományok számos területén, így a fizikában, a kémiában és a biológiában is elfogadottá vált. A humán tudományok közül a közgazdaságtan, a szociológia és a pszichológia témakörében gyakori az alkalmazása. Ezen kívül a szimulációnak komoly szerepe van a mérnöki tudományokban is, ahol a rendszer működésébe nyerhetünk bepillantást általa.

A szimulációkkal elért eredmények tudományos értékét nagymértékben meghatározza a módszer alkalmazásának alaposssága. Mivel szimuláció esetén a modellezett jelenség számos irreleváns(nak vélt) tulajdonságától eltekintünk, részben a probléma áttekinthetősége, részben a számítási kapacitás végessége miatt, ezért a konklúziók minél alaposabb tesztelése szükséges.

A validálásnak különböző szintjei léteznek, mint ahogy arra Gulyás ([65]) is rámutat. Először is az eredményeket a szimulációs környezetben kell a paraméterek átfogó alakításával ellenőrizni. Ezzel egyrészt a megfogalmazott állítások érvényességi köre is meghatározható, másrészt a véletlen eseményekből adódó eltérésekkel szembeni robusztusság is igazolható. Utóbbi érdekében véletlenszámgenerátorokat érdemes alkalmazni, melyek különféle kezdeti értékkel elindítva különböző pszeudo-véletlen eseménysorozatokat hoznak létre.

Az eredmények tesztelésének következő szintje az áttérés laboratóriumi környezetre. Ebben az esetben a vizsgált jelenség kikerül a számítógép virtuális világából a fizikai környezetbe, de annak még csupán egy sterilebb, ingerszegényebb változatába, ahol esetleg egy mesterséges környezetben lehet a problémát vizsgálni. A tesztelés ezen szintje különösen robotikai kísérletekben jellemző (1.1. ábra).

Az eredmények igazán megnyugtató igazolása a valódi, természetes környezetben végzett teszteléssel jön el. Ekkor a szimulációban meglévő egyszerűsítések mind eltűnnek, és kiderül, hogy a számításon kívül hagyott részletek, a környezet nem modellezett elemei valóban nem érintik az eredmények érvényességét.



1.1. ábra. LEGO robot kísérleti terepen. A 2002-es 24 órás programozói verseny egyik résztvevője fekete-fehér vonalak segítségével tájékozódik.

A tesztelés legutolsó szintje azt vizsgálja, hogy az elemzett jelenség más értelmezési tartományokon megfigyelhető-e, azaz mennyire létezik környezetfüggetlenül. Amennyiben több szimulációs eredményt sikerül ilyen módon összekapcsolni és igazolni, akkor az mélyebb összefüggésekre utal.

A fejezet további részében áttekintem a vizsgálataim szempontjából kiemelt jelentőségű ágensalapú szimulációt és annak fajtáit, kitérve a szimulációból adódó előnyökre és hátrányokra. Végül röviden bemutatom a Webots robotszimulációs és a Repast általános ágensszimulációs környezeteket, melyeket kísérleteimben használtam.

1.2. Ágensalapú szimuláció

Az ágensalapú szimuláció a szimuláció mint tudományos módszer egy viszonylag új ága, melyet részben vagy teljesen átfedő egyéb elnevezésekkel is illetnek, így ágensalapú modellezés, ágensalapú számítás, egyén alapú szimuláció ([43]).

Ezen módszerek közös sajátsága, hogy komplex rendszerek kezelésére alkalmasak, olyanokéra, ahol valamilyen környezetbe ágyazott, többé-kevésbé intelligens viselkedést mutató ágensek testesítik meg a vizsgálat tárgyát.

A terület igen széles irodalma és az alkalmazások nagy száma ellenére az ágens fogalma nem teljesen tisztázott, a közösnek tekinthető álláspont szerint az ágensalapú rendszerek szereplőinek alapvető tulajdonságai az autonómia, a reaktivitás, a proaktivitás és a célvezéreltség. Emellett számtalan egyéb hasznos tulaj-

donságot lehet megemlíteni, például a racionális vagy emberi viselkedést[1], a mobilitást, a megbízhatóságot, a kommunikáció lehetőségét, a kooperativitást, a tanulás vagy tágabb értelemben az adaptáció képességét ([198], [82]). Az alapvető fogalmi kérdések körüli tisztázatlanság teljesen analóg a mesterséges intelligenciát érintő nehézségekkel. Mégis, a részleteiben eltérő felfogást képviselő ágensek jól alkalmazhatók minden olyan területen, ahol a feladat környezete kellően komplex, bizonytalan, esetleg változó, vagy amikor az ágens egy természetes metafora, illetve ha az adatok, az irányítás vagy a szakértelem a környezetben megosztott módon van jelen. Az ilyen tulajdonságú feladatok a légiforgalom-irányítástól és a katasztrófavédelemtől, az élőlények, a robotok modellezésén át az üzleti és ipari rendszerek kezeléséig mindenütt megtalálhatók, így érthető módon az ágensalapú hozzáállás igen elterjedt. Gyakorta használnak ágenseket a részvételi szimulációk során, amikor a kísérleti személy az ágensrendszer egyik egyedének szerepébe kerülve egyszerre szemlélője és cselekvője a szimulált világnak, és mintegy belülről gyűjti tapasztalatait ([77]).

Az ágensalapú modellezés elméletének a gyakorlatba való átültetése, az ágensarchitektúrák létrehozása során a fogalmi alapozáshoz hasonló módszertani bőséggel találkozhatunk. Nagyon sok ágensalapú rendszer készült és készül, s ezek egyúttal lényegesen eltérő megközelítéseket is képviselnek ([199]), mint ahogy az a következőkben is látszik.

A klasszikus mesterséges intelligencia a gondolkodást mint a szimbólumokon és az azokból képződő/képezhető struktúrákon végzett komputációs tevékenységet vizsgálta ([124]). Az ennek megfelelő rendszerek tipikus példája, a STRIPS szimbolikus világ- és cselekvésleírásokkal készít tervet ([55]).

A hetvenes évek második felétől terjedt el a konnekcionista architektúra, mely a szimbólumok helyett az idegrendszerszerű modellálást helyezte előtérbe ([138]). A gyakorlatiasabb megközelítés mérnöki szempontból volt vonzó, a kisebb elemekre alapozott, hálózatszerű felépítés nagyobb flexibilitást és párhuzamos végrehajtást tesz lehetővé ([83]).

A szimbólumfeldolgozással szembeni másik alternatív megközelítés elsősorban Brooks nevéhez fűződik, aki a világ teljes részletességű modellezését és a hozzá kapcsolódó tervgenerálást kivitelezhetetlennek, ugyanakkor szükségtelennek tartja, ehelyett egy viselkedésalapú rendszert javasol ([26]). Szerinte az intel-

[1] A két eltérő megközelítés mesterséges intelligenciabeli szerepéről bővebben [59] ír.

ligens viselkedés reaktív módon, explicit szimbolikus reprezentáció és következtetés nélkül is megvalósítható valódi, fizikai ágensekkel. Ezek az ágensek szituált és megtestesült módon dolgoznak, a világot önnön modelljeként használva. Az intelligencia a komplex rendszer elemeinek összhangjából emergens módon alakul ki ([22], [23], [24]).

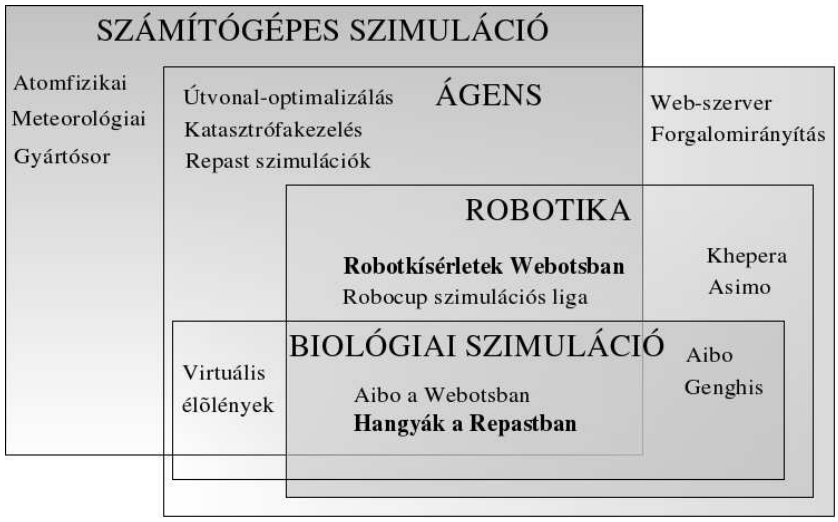
A teljes modellt és a teljes modellnélküliséget kínáló két megközelítést ötvöző hibrid architektúrák a nyolcvanas évek végén jelentek meg. Ezek egyik jellemző képviselője a Ferguson által készített TouringMachines, mely reaktív, tervező és modellező rétegek párhuzamos működtetésével, a flexibilitás biztosítása mellett a belső állapot fenntartását is lehetővé teszi ([54]).

Az ágenseknek az intelligencia modellezésén túl is lehet szerepük. Az ágensalapú modellezést hatékony informatikai rendszer létrehozásának nézőpontjából vizsgálva látható, hogy mint új programozási paradigma is megállja a helyét. Az egyre összetettebb információalapú világban egyre inkább szükség lesz olyan szoftverrendszerekre és -komponensekre, melyek nyitottak, decentralizáltak, szituáltak, valamint csupán lokális kontrollt és lokális kapcsolatokat igényelnek, ahogy arra Zambonelli és Parunak rámutat ([201]). Az ágensek a komplex rendszerek elemi összetevőinek természetesebb modelljei, mint az eddigi megközelítések, így velük könnyebb a rendszert megtervezni, illetve az így készült rendszer jobban áttekinthető.

Az ágensalapú modellezést összevetve a jelenleg leginkább elterjedt objektumorientált paradigmával ([132]) sok hasonlóság fölfedezhető, ugyanakkor az ágensalapú megközelítést több értelemben is továbblépésnek tekinti többek között Jennings ([80]). Míg az objektumok passzívak, addig az ágensek aktívak, sőt proaktívak, vagyis bennük nem csupán a viselkedés leírása, hanem aktiválása is megvalósul. Ezen túlmenően az objektumok a világ túlságosan elemi felbontását adják, ami a tervezési minták, a különféle magasabb szintű keretrendszerek nagy számában is tükröződik ([131]). Végül, az objektumok közötti rendezettséget leíró öröklődésnél sokkal gazdagabb kapcsolati hálóra van szükség és lehetőség az ágensek esetén.

Az eddigiek összegzéseként elmondható, hogy az ágens alapú szimuláció mind az informatika, mind a mesterséges intelligencia szempontjából egy ígéretes terület, mely hatékony módszereket és új nézőpontot hozhat a kutatásba. A fejezet fogalmainak kapcsolataira világít rá az 1.2. ábra.

Dolgozatom további részében ágensalapú szimulációkat mutatok be, melyek



1.2. ábra. A szimuláció, az ágensek és a robotika kapcsolata néhány példán keresztül. A dolgozatban alaposabban vizsgált két terület ki van emelve.

a biológiai és a robotikai szimulációk témaköréből kerülnek ki, így az ágensalapú megközelítés természetes metafora. A vizsgált témakörök az ágensarchitektúrák eltérő fajtáit alkalmazzák, a robotikai verseny és a hangyák esetén a világmodell nélküli reaktív kontrollereknek jut szerep, míg a térképépítést végző robotokat inkább a hibrid rendszerek körébe lehet sorolni. Mégis az elvégzett kutatások közös jellemzője az ágensek navigációs képességének vizsgálata.

1.2.1. Biológiai alapú szimuláció

Az ágensalapú szimulációk egy érdekes típusa a biológiai alapú szimuláció. Az ezzel kapcsolatos kísérletek két téma köré csoportosulnak. Makroszinten azt érdemes vizsgálni, hogy *mit* csinálnak az élőlények: milyen viselkedések jellemzik őket, és ezt miként lehet modellezni. Ezek a kérdések a szimulációt az etológiához ([41], [91]), emberek esetén a szociológiához kapcsolják ([36]). Mikroszinten a kísérletek célja annak vizsgálata, hogy a viselkedések *hogyan* valósulnak meg: a háttérben milyen biológiai struktúrák, funkciók, mentális folyamatok húzódnak meg. Ez a terület a szimuláció neurobiológiával való kapcsolatát erősíti ([3]). A biológiai rendszerek megismerésén túl a vizsgálatok bevallott célja az is, hogy a természet jól bevált, hosszas evolúció által kiérlelt módszereit ellesve hatékony algoritmusokat vagy teljes rendszereket (robotokat) hozzanak létre ([189], [168]).

A biológiai alapú ágensszimulációkat a nyolcvanas évek közepétől kezdve jel-

lemzően áthatja az *animat*[2] fogalma, melyet Wilson vezetett be, egyaránt alkalmazva azt szimulált élőlényekre és élőlényt utánzó robotokra ([197]). Az intelligencia építésének animatok által kijelölt útját nem a hagyományos analitikus, a rendszert fentről lefelé vizsgáló módszerek jellemzik, hanem szintetikus, komputációs, alulról felfelé építkező látásmódot képviselnek. Turing jóval korábbi megfogalmazásában ez egy gyermekgép létrehozását jelenti, melyet valódi környezetbe helyezve, tapasztalati úton tanítva lehetne megalkotni egy komplex, holisztikus rendszert ([191]). Az animat megközelítés erejét az adja, hogy a mesterséges élőlény önfenntartási (evés, pihenés), illetve fajfenntartási ösztöne és egyéb igényei (játék, felfedezés) formálhatják a fejlődést, és nincs szükség tervezett beavatkozásra.

Az animatok egyik első képviselője Brooks, aki a korábban említett viselkedésalapú rendszert következetesen alkalmazta egy hat lábon járó robotcsótány mozgáskoordinációjának megvalósításához ([22]). A kísérlet során kiterjesztett véges állapotú automatákból álló elemi viselkedésmodulok egymásra hatásából alakulnak ki a magas szintűnek tekinthető cselekvések. A tervezés és fejlesztés során a modulok egymás után épülnek be a kialakuló animatba, mindig működőképes egyedet kialakítva, biztosítva az egyes modulok kellően robusztus megvalósítását.

Maes a viselkedésalapú rendszert járás helyett az animat teljes viselkedésrepertoárjának meghatározására alkalmazza ([104]). A viselkedés az önfenntartás néhány modulja (pl. evés, menekülés, alvás) egymásra hatásának eredménye. A modulok végrehajtása az aktivációs szint aktivációs küszöbhöz viszonyított értékétől függ. Az aktivációs szintet közvetlenül a motivációk, közvetve a többi viselkedés befolyásolja megerősítő és gátló kapcsolatokon keresztül. A központi irányítás nélküli rendszerből életszerű viselkedés bontakozik ki.

Az emergencia — azaz összetett mintázat kialakulása olyan elemiekből, melyek azt közvetlenül nem kódolják — egy újabb példája Reynolds madárraj jellegű viselkedést mutató multiágens kísérlete ([146]). A csoport animatjait irányító szabályok rendkívül egyszerűek: távolságtartás a környező tárgyaktól, társaktól, a többiek sebességéhez idomulás, a csoport tömegközéppontjához tartás. Ennek eredményeként a véletlenszerűen mozgó animatok központi tervezés nélkül is hamar „madárrajjá” állnak össze, melyet a környezet akadályai is csak időlegesen tudnak szétválasztani. Elemi szabályok egymásra hatásából kibontakozó komplex

[2] A szó az artificial animal és az automaton kifejezések összevonásával keletkezett.

viselkedés más multiágens-szimulációnál is megfigyelhető, így ételgyűjtő hangyák útvonal-optimalizálása ([50]), különféle lárvaválogatása ([46], [196]), termeszvárépítés ([181]) is sikeresen leírható emergens modellekkel.

1.2.2. Robotok szimulációja

Russell és Norvig szerint a mesterséges intelligencia egyik legnehezebb és talán legérdekesebb feladata intelligens robotok létrehozása ([150]). Gyakorlatilag a mesterséges intelligencia legtöbb területének eredményeit kell benne ötvözni, miközben a feladat maga szinte minden kategorizálás szerint a legnehezebb választás: folytonos térben és folytonos időben, nem determinisztikus, zajos, nem hozzáférhető környezetben, változó körülmények között kell a robotoknak működni.

A megoldáshoz vezető úton a feladat különféle egyszerűsítése is elképzelhető. A sikeres tudományos kísérletek is az egyetemi folyosók vagy múzeumok bejárásáról számolnak be, a robotok és tervezőik ritkán merészkednek ki a szabadba. Vagyis a mai robotok modellvilágokban működnek, és ezen modellvilágokról építik fel saját belső reprezentációikat.

Az elv egy lehetséges továbbvitele a nagy kapacitású, olcsó számítógépek elterjedésével egyszerűen adódik: miért ne lehetne a robotokat virtuális világokban működtetni, azaz speciális robotszimulációs szoftvereket használni? Így először a valós feladatok megoldása szempontjából nem megfelelő, kifejezetten egyszerű virtuális világok felé indulunk el, de reméljük, hogy ez a visszalépés később gyorsabb haladást tesz lehetővé. Ez a megközelítés az *intelligencia másodfajú modellezésének* is tekinthető: az emberi intelligenciát részben modellező robotok modellezhetők a szimulátorokban ([174]).

Szimulátort használva a kutató saját elképzeléseinek megfelelően építhet fel kísérleti környezeteket, melyek bonyolultságát, részletgazdagságát, valósághűségét növelheti egészen addig, míg a virtuális robotok digitális bölcsőjükből végre zökkenőmentesen kimerészkedhetnek az igazán komoly kihívásokat jelentő fizikai valóságba.

Az elmúlt évtizedig a robotszimulátorok többnyire ipari alkalmazásokban bukkantak fel — pl. az Adept[3] Digital Workcell nevű gépsorszimulátora —, illetve egy adott tudományos kísérlet igényeinek megfelelő ad hoc szimulátort ([90],

[3] http://www.adept.com/

[96]) vagy speciálisan az alkalmazott robothoz kialakított szimulátort használtak ([74]). Az 1.3. és az 1.4. ábra is egy ipari járművet és a hozzá tartozó célszimulációt mutatja be.

1.3. ábra. A finn PlusTech hatlábú robotja, mely fák begyűjtését végzi.

1.4. ábra. A robotcsalád kereken guruló változatának szimulációja.

Az utóbbi évtizedben a szimulációk az általános felhasználhatóság irányába mozdultak el. A népszerű robotfutball[4] szimulációs ligája sem robotspecifikus, egy általános kísérletező szoftvert jelent a multiágens területen dolgozóknak, melynek egyes eredményei átvihetők a valódi robotos ligákba is ([85]). Az MIT Leg laboratóriumából kinőtt Yobotics[5] mindenféle lábbal rendelkező robotok és protézisek szimulációjára összpontosít. Az ingyenesen elérhető Player/Stage szimulátor[6] tucatnyi robottípus programozását teszi lehetővé sokféle nyelven, további bővítéseknek is teret hagyva ([61]). Az újabb szimulátorokkal az eredmények könnyebben megoszthatók, ellenőrizhetők, jobb kiindulási alapot jelenthetnek a valódi robotos kísérletekhez.

A külvilág és a robot mesterséges felépítésével sok kötöttségtől lehet megszabadulni, ugyanakkor újabb problémák is jelentkezhetnek. A szimuláció előnyeit és hátrányait tekinti át a következő rész.

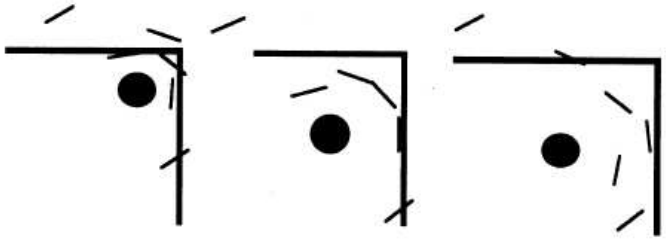
A szimuláció hátrányai

A szimuláció egyik legnyilvánvalóbb hátránya, hogy plusz számítási költségként jelentkezik a fizikai világ által egyébként „ingyen” adott háromdimenziós tér megalkotása, az érzetek felépítése, a mozgás kinematikája, dinamikája. A szimulátornak kell kiszámítania a robot érzékszervein megjelenő információt, látás esetén

[4] http://www.robocup.org
[5] http://yobotics.com
[6] http://playerstage.sourceforge.net/

például az árnyékolást, a takarásokat, a tükröződést figyelembe véve. További feladata, hogy a mozgás során a robot felépítéséből adódó megszorításoknak megfelelő hiteles útvonalat kell előállítania, amely a nem várt események, például ütközések eredményét is magában foglalja.

Komoly veszélyt jelenthet az, hogy esetleg a szimuláció és a valóság közötti eltérések miatt a szimulált robot ugyan jól végzi feladatát, a valóságban teljesen használhatatlan. Ennek elkerüléséhez nagyon alaposan kell meghatározni a világ modellezendő jelenségeit, ahogy arról Brooks ([25]) és Jakobi ([79]) is ír.



1.5. ábra. Szonáron alapuló távolságmérés hibája. A sarok közelében lévő kör alakú robot méréseit a rövid vonalak jelzik, melyek láthatóan nem esnek egybe a fal vonalával ([109], M. Mataric engedélyével).

Érdemes figyelembe venni, hogy nincs két teljesen egyforma alkatrész, így két motor nem egyforma erővel gyorsít, két kamera kissé eltérő színeket ad vissza ([127]). Fontos modellezendő tulajdonság az érzékszervek és a cselekvések már említett zajossága, bizonytalansága. A robot kereke a gyenge tapadás miatt megcsúszhat, sárban elakadhat, szőnyegen, parkettán ugyanazon hajtóerőnél is eltérő sebességgel haladhat, sőt még akár a szőnyeg száliránya is számíthat. Egy szonár, azaz távolságérzékelő által kibocsátott jel különféle felületekről különböző módon verődik vissza, eltérő információt közvetítve. A visszaverődések miatt akár másik szonárhoz is eljuthat a jel, ami nyilvánvalóan hibás mérést jelent. Fontos lehet a jel beesési szöge is: az 1.5. ábrán látható, amint a kör alakú robot egy sarkot közelít meg, és szonárokat használ fel a távolság becslésére. A mérések megközelítőleg pontosak merőleges beesésnél, de a nagyobb szögeltérések esetén láthatóan komoly hiba adódhat ([109]). Ebből is látszik, hogy az érzékszerveket a valóságban is kalibrálni kell, melynek valahogy a szimulátorban is meg kell jelennie.

Az érem másik oldala, hogy egy szimuláció olyan problémákat is fölvethet,

mely a valóságban nem jelentkezik. Egy egyszerű példa robotok szembetalálkozása egy négyzetrácsszerű világban. Ilyen esetben a szimulátornak megoldást kell találnia a holtpontos helyzet feloldására. Bár természetesen a valódi robotok is ütközhetnek, az apró eltérések miatt nem fognak pontosan egyszerre, pontosan ugyanabba a pozícióba érkezni, így a jelenség igazából nem is megoldandó ([25]).

A szimuláció előnyei

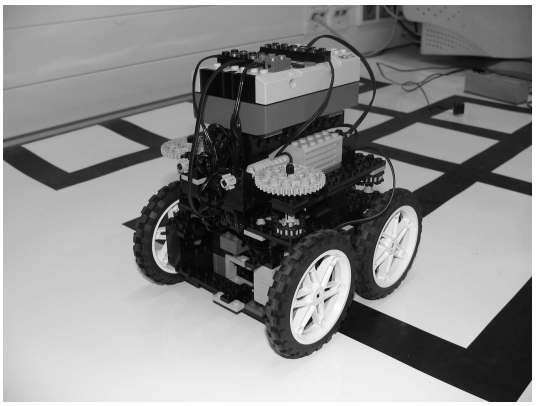
Ha ennyi baj van a szimulációval, akkor miért érdemes mégis használni? A szimuláció egyik legfontosabb jellemzője, hogy a kísérlet minden paraméterét a kutató maga kontrollálhatja. Ezzel egy adott környezet kissé eltérő feltételei között vizsgálhatja a robot működését, így például a különböző módon csúszó padlók hatását elemezheti. Nagyobb léptékben a robot lényegesen eltérő világokban bizonyíthatja képességeit, vagyis teljesítménye a kutatólaborok mellett városi forgalomban, erdőben vagy jégmezőn is vizsgálható. Ez egyúttal a létrehozott algoritmusok robusztusságát is mérhetővé teszi a kisebb-nagyobb változásokkal szemben.

A szimuláció használatának egy kevésbé tudományos oka, hogy olcsóbb. Nem kell költeni a robotra, multiágens kísérletek esetén robotokra, a különféle érzékszervekre vagy a fizikai, de mégis valamennyire mesterséges környezet kialakítására. A sokrobotos kísérlet technikai nehézséget is jelenthet, az irányításért és az energiaellátásért felelős kábelek összegabalyodhatnak. Ha pedig nincsen kábel, akkor gondoskodni kell a robotok akkumulátorának időnkénti feltöltéséről ([107]).

Szintén technikai jellegzetesség, mégis a hatékonyságot növelheti a szimuláció párhuzamos végrehajtása: ekkor a kutatók a kísérletet különféle paraméterekkel végzik el egyszerre, szemben egy robot egymás utáni feladatmegoldásaival. A párhuzamosítás a különféle feladatkiírások közül a kutatás szempontjából érdekesek kiválasztásában is segíthet.

A szimulátor valamennyire idealizált világa kiküszöbölheti azokat a technikai problémákat, melyek kívül esnek a feladat specifikációján, így az érzékelők, a meghajtás és a robot bármilyen mozgó alkatrészének műszaki hibáival. Az 1.6. ábrán bemutatott robot is kellően bonyolult, és sok alkatrészből áll, így az üzemzavarok a valódi fejlesztést lassítják ([76]).

Hasonlóan fontos szempont, hogy a szimulátorban gyorsabban lehet létrehozni egy kísérletet, mint a valóságban ([113]). Ez akár igazi robotokkal való



1.6. ábra. Egy LEGO robot az ELTE robotikaórájáról (Istenes Z. engedélyével).

feladatmegoldásnál is hasznos, mint a probléma prototípusának prompt létrehozása. Szimulátorban a kísérlet összeállítása — a környezet felépítése és a robot kialakítása — mellett a kísérlet végrehajtása is lényegesen gyorsabb lehet. Ez akár egyes futások elemzésekor is fontos, de különösen előnyös például induktív tanulási eljárások esetén, amikor általában nagyszámú példa alapján fejlődik az irányító program. Így például neurális hálóknál a valós robot csupán előzetesen összegyűjtött adatokon végezhet többnyire off-line tanulást, hiszen nem mehet végig a kísérleti környezeten akár több ezerszer, az időigény és az alkatrészek kopása miatt sem. A számítógép belsejében online tanulásra is mód nyílik. A probléma még nyilvánvalóbb evolúciós algoritmusok esetében. Ekkor egy népes robotpopuláció sok generációjának sok egyedén kell sok kísérletet elvégezni, hogy a robotok egyedi életrevalóságát meg lehessen határozni. Így az új robotgeneráció életképesebb, az adott feladatot ügyesebben megoldó példányokat tartalmaz. Ez csak kevés valós robottal lehetséges, ellentétben a szimulátorral ([79]).

A szimulátor ezen kívül segítheti a robotirányító eljárás mellett a felépítés kialakítását is. Variálható a robot alkatrészeinek típusa, száma, elhelyezkedése, a hajtási mód, a szenzorok felbontása stb., hasonlóan ahhoz, ahogy Sims virtuális élőlényei fejlődnek ([158], [159]).

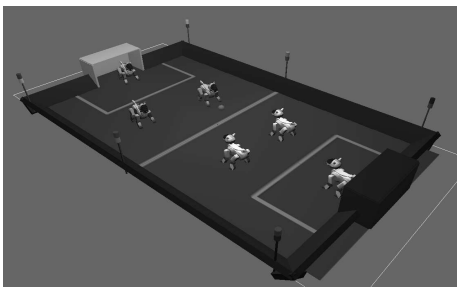
A szimuláció és a valóság közötti eltérésekből adódó kihívásra pedig a szimulált robotok kódjának és magának a szimulátornak a koevolúciója adhat választ, melynek célja a külvilág a kísérlet szempontjából minél hűségesebb modellezése ([25]).

1.3. A használt szimulációs környezetek

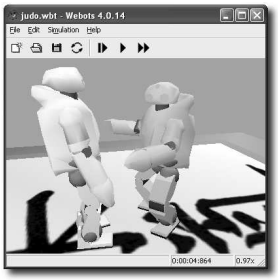
Kutatásaim során a Webots robotszimulációs és a Repast multiágens-szimulációs környezeteket használtam.

1.3.1. A Webots szimulátor

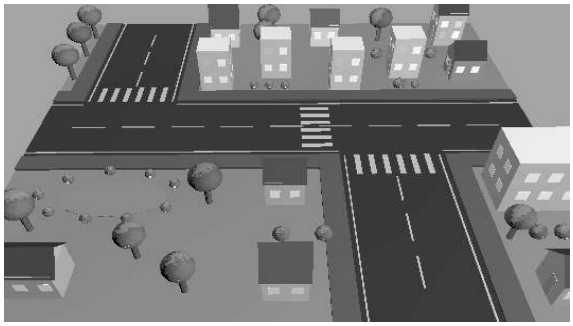
A Webots mobilrobot-szimulátor a Cyberbotics által fejlesztett kereskedelmi termék[7] ([111]), ami egy nyílt forráskódú Khepera robotot szimuláló csomagból fejlődött ki ([112]).



1.7. ábra. Futballozó Aibo kutyák a Webots szimulációs környezetben. A robotfutball-világbajnokság négylábú ligájának szimulált változata.



1.8. ábra. Sony Qriok harcolnak a Robot Judo Contesten, ahol az ellenfél robotját kellett 5 másodpercre leszorítani vagy a játéktérről kilökni.



1.9. ábra. Egy városi környezet forgalomoptimalizálási kísérletekhez.

A Webots képes kereken guruló, gyalogló vagy akár repülő robotok különféle típusainak kezelésére, miközben valódi robotok (pl. Pioneer, LEGO Mindstorms, Aibo) irányítására is alkalmas ([113]). A háromdimenziós kísérleti környezetek VRML nyelven írhatók le, a robotokat irányító kódot több magas szintű programozási nyelven is meg lehet adni. A robothoz eljutó érzetek különféle modalitású

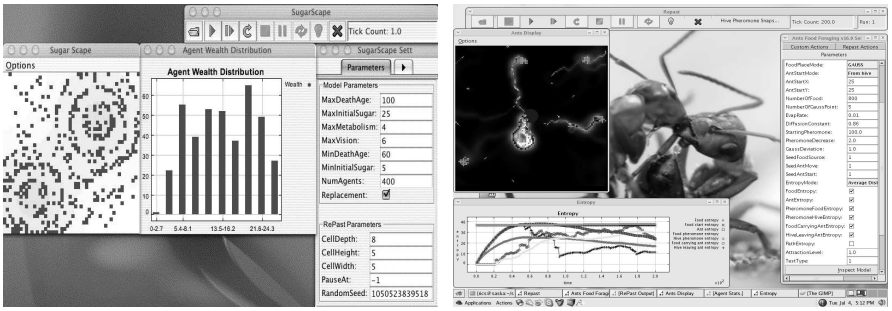
[7] http://www.cyberbotics.com

szenzoroktól származnak: tapintás-, fény-, távolság-, elfordulásszenzor, valamint kamera tartozik a lehetőségek közé. A robotok mozgását a kinematika törvényei határozzák meg.[8] A környezet előnyös tulajdonsága, hogy viszonylag egyszerűen lehet új világokat, robotokat létrehozni és tesztelni benne. Az 1.7., az 1.8. és az 1.9. ábra néhány, a Webots környezetben elkészült szimulációt mutat be.

1.3.2. A Repast szimulátor

A Repast[9] a chicagói egyetemen készített nyílt forráskódú, Java nyelvű multiágens-szimulációs programcsomag ([33]).

A Repast alapvető koncepcióit a Swarm[10] ágensszimulációs környezetből kölcsönözte ([114]), de túllép azon, többek között az ágensek közötti kapcsolatok és a térbeli elhelyezkedés definiálásának sokféleségével, illetve neurális hálók, genetikus algoritmusok, regresszió felhasználásának támogatásával ([185]). A diszkrét térben és időben, sztochasztikus módon végzett futásokat grafikusan és kötegelve is el lehet végezni. Az eredmények kiértékeléséhez komoly segítséget jelent a beépített naplózás és grafikonkészítés lehetősége. Repastban készült szimulációk láthatóak az 1.10. ábrán[11].



(a) A SugarScape kísérlet (b) Hangyák szimulációja

1.10. ábra. A Repast szimulátor. Az ablakok a szimuláció kezelőfelületét, az aktuális paramétereket, a kísérlet kétdimenziós állapotterét és az elemző ablakokat mutatják.

[8] A legújabb változatokban már létezik támogatás erőket modellező eljáráskönyvtárakhoz.

[9] Recursive Porous Agent Simulation Toolkit – http://repast.sourceforge.net

[10] http://www.swarm.org

[11] A SugarScape kísérletben újratermelődő cukor gyűjtése a feladat, az éhen maradók elpusztulnak.

2

A navigáció különféle módszerei

2.1. A navigáció nehézségei

Képzeljük el, hogy egy ismeretlen város közepén tesznek ki bennünket egy papírral és egy ceruzával, térkép nélkül, azzal a feladattal, hogy megismerjük környezetünket! Ahogy járkálunk, lejegyezzük az épületek hozzánk és egymáshoz viszonyított helyzetét. Saját pozíciónkat igyekszünk kiindulási pontunkhoz, valamint a térkép eddigi elemeihez kapcsolni. Vagyis pozíciónktól függ a térképi elemek elhelyezkedése, mi pedig a térkép alapján határozzuk meg pozíciónkat, a térkép segítségével navigálunk. Egy mobil ágensnek, legyen az élőlény vagy robot, ismeretlen környezetben szintén meg kell küzdenie ezzel a tyúk-tojás jellegű feladattal, ami pontos, használható térképet csak hosszú idő után eredményez ([162]).

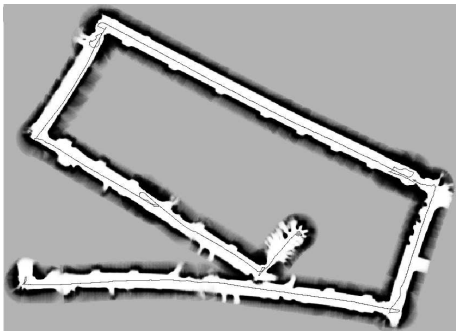
A szakirodalomban *szimultán lokalizáció és térképezés*[1] névvel illetett probléma mellett egyéb, komoly nehézségek is adódnak.

Az érzékelt környezet és a válaszként adott cselekvések pontatlanok, hibásak. Ez biológiai és mesterséges ágenseknél is így van. Az élőlények intencionális hozzáállást alkalmazva próbálják megjósolni környezetük viselkedését, ami annyira lesz helyes, amennyire az a túléléshez feltétlenül szükséges ([47]).

A mesterséges ágensek esetén a radarok, kamerák tévednek, a göröngyös talaj vagy a szőnyeg széle problémákat okoz. Ez természetesen megnehezíti a térképé-

[1] Kétféle angol elnevezés is közismert: Simultaneous localization and mapping (SLAM) [49] és Concurrent mapping and localization (CML) [97]

pítést is. A probléma igazi nehézségét az adja, hogy a tájékozódás közben fellépő zaj nem független az ágens cselekedeteitől. Ha így volna, akkor a mérések egy idő után konvergálnának a helyes értékekhez. Azonban a mérési hibák akkumulálódnak, vagyis nagyobb távolságok megtétele után, ha nincsen valamilyen viszonyítási pont, akkor a pozícióbecslés komolyan eltér a helyes értéktől ([183]). A 2.1. ábra azt mutatja, hogy egy 20x60 méteres téglalap alaprajzú folyosót körbejárva egy robot az összegyűlt hibák miatt mennyire torz térképet készít.



2.1. ábra. A mozgás közben fellépő zaj szerepe. A téglalap alakú szobát megkerülő robot, az egyenes szakaszokon, de különösen a 90 fokos fordulóknál akkora szögeltérést gyűjtött össze, hogy a térkép alsó részén lévő folyosó megkettőződve jelenik meg ([182], S. Thrun engedélyével).

A minél hatékonyabb navigációhoz pontos térkép szükséges. A részletesség növelése többnyire a számításigényt növeli. A környezet magas szintű, objektumalapú leírása néhány tucat elemmel (falak, ajtók, folyosók) megtehető. Egy részletes kétdimenziós térkép néhány ezer entitást tartalmaz, míg egy teljes háromdimenziós modell a többmilliós elemszámot is elérheti. Ha a hatékonyság érdekében egy ágens mégis csak kevés információt használ fel a környezetéből, akkor könnyen ütközhet az *adatösszerendelés*[2] problémájába ([125], [99]). Ez akkor merül föl, ha két hasonló helyről – esetleg különböző nézőpontokból – nem könnyű eldönteni, hogy megegyeznek-e. Ilyenkor egy körfolyosó bejárása után nem lehet észrevenni, hogy visszaértünk-e a kiindulási pontra. Ez azért alakul így, mert a használt szenzorok által visszaadott értékek gyakran nem egyediek a kísérleti környezetben. Az embereknél a fejlett látás miatt ritkán fordul elő ez a probléma, például egy növénylabirintusban. A jelenségtől megvédhet a *szenzoriális fúzió*[3], mely az – olykor lényegesen különböző típusú – érzékelők ál-

[2] data correspondence
[3] sensor fusion

tal egyszerre összegyűjtött információ egyesítésével egyre pontosabb és egyedibb adatokat biztosíthat ([117], [148]).

A sikeres ágenseknek még egy elég fontos problémát kell megoldaniuk. A térkép elkészítése után mozgástervezést kell végezniük, aminek segítségével a terepviszonyoknak és mozgási korlátaiknak megfelelő útvonalat találnak céljuk felé ([155]).

A navigáció további nehézsége a változásban keresendő. Az élőlények viselkedését befolyásoló természetes környezet változása mellett a robotok munkakörnyezete is átalakul: emberek járkálnak körülöttük, ajtók nyílnak-csukódnak, helyiségek átrendeződnek. Ennek a változásnak természetesen a térképen is tükröződni kell, vagyis a megoldás egy újabb dimenzióját az idő adja. A környezet különböző elemei eltérő sebességgel és eltérő módon változnak. Nem egyforma modellt igényelnek az éves és a napi ritmusok, illetve a térképen szereplő ember (gyakran mozog), ajtó (gyakran mozog, de helyhez kötött), szekrény (ritkán mozog) vagy fal (egyáltalán nem mozog).

Igazán aktívan kutatott terület a szabadban végzett navigációé, hiszen egyrészt a tényleges feladatok jó részét nem épületekben kell végezni, másrészt a navigáció épületen belül alkalmazható, struktúrákra építő módszerei nem mindig vihetők ki nyílt térbe, tehát újabb megoldások is szükségesek ([183]). A jelenlegi algoritmusok jól körülhatárolt tárgyak, jellemző mintázatok, színek érzékelésére vannak felkészülve, ezért szélfújta, sűrű erdőben, illetve sivár homok- vagy jégsivatagban nehezen találnak tereptárgyakat. Ez a távolságok mérését is megnehezíti, a fák lombozatáról a jelvisszaverődés rendkívül esetleges, míg kietlen terepen csupán távoli kiemelkedések jelenthetnek fogódzót. A robot mozgástervezését és -kivitelezését is merőben más szabályok szerint kell elvégezni, hisz az erdő sűrű aljnövényzete, egy összeomlott, néha megmozduló épület, jeges vagy sáros felszín, víz alatti áramlások mind komoly nehézségeket okozhatnak. Láthatóan ezek a környezetek jóval nagyobb változatosságot is mutatnak, mint az az eddigi kísérletezéseknél megszokott.

Az eddigiek összefoglalásául elmondható, hogy egy intelligens ágensnek a fenti feladatok autonóm, robusztus megoldását akár tetszőleges, ismeretlen környezetben is valós időben kell elvégeznie, lehetőleg univerzális módon. Jelenleg ez a mobil robotika egyik legfontosabb, egyelőre csak részlegesen megoldott problémája ([26]). De ez csupán a kezdet. Az igazi feladat — takarítás, fűnyírás, autóvezetés, házépítés vagy holdjárás — csak ezek után következik.

A fejezet hátralevő részében bemutatok néhány példát, melyek a vázolt problémákra adnak többnyire részleges választ, áttekintem, hogy egy teljes navigációs rendszer milyen elemekből épül föl, és hogy ezek a részegységek milyen alternatív megoldások közül kerülhetnek ki.

2.2. Biológiai alapú navigáció

Ahogyan az a többi mérnöki tudománynál is jellemző, a robotika is előszeretettel alkalmaz az élővilágból megismert megoldásokat, hiszen így az evolúciós verseny által optimalizált eszközhöz vagy eljáráshoz lehet jutni. Ugyanakkor a navigáció biológiai alapú megoldásai gyakran egyáltalán nem követik azt a felépítést, melyet egy mérnöki megközelítés adna, emiatt alkalmazásuk egyáltalán nem triviális, ahogy arra Franz és Mallott is rámutat ([57]). A tájékozódás biológiai folyamatainak hátterében meghúzódó mechanizmusok megismerésében kiemelt jelentősége volt Tolman munkájának, mely a korszak szélsőségesen redukcionista nézetével szemben állította, hogy létezik az élőlényeknek belső állapota. Ő ezt rágcsálókkal végzett labirintusos kísérleteire alapozta, ezek eredményeként vezette be a *kognitív térkép* fogalmát ([186]). A kognitív térkép hagyományos felülnézeti, szimbólumokkal ellátott térképeinktől eltérő mindazon reprezentációk összessége, mely az élőlény számára fontos helyek (ételforrás, búvóhely, fajtársak) memorizálását lehetővé teszi.

A navigáció különböző felépítésű és bonyolultságú térképekre épül, és így az élőlények fejlettségi szintjétől és életmódjától függően eltérő lehet ([41]). A legegyszerűbb orientációs mód a kinézis, melynél az állat az inger nagyságával arányos mozgási reakciót ad, az inger irányát még nem veszi figyelembe. Ilyen egyszerű viselkedés már a papucsállatkánál is megfigyelhető. A taxisok különböző, egyre fejlettebb formáinak (klinotaxis, tropotaxis, telotaxis) használatakor az egy vagy több érzékszervre érkező ingereket hasonlítják össze, és az eltérésből számítják ki új haladási irányukat különféle rovarok, rákok. Ilyenkor a célt vagy egy azt azonosító jelet folyamatosan érzékel az élőlény. Ezeket a viselkedéseket jól modellezik a következő fejezetben ismertetett Braitenberg-járművek is.

A biomimetikus navigáció egyik legkorábbi magyar vonatkozású példája a Szegedi Katica, Muszka Dániel alkotása 1957-ből, Kalmár László tervei alapján ([15]). A szerkezet alapvetően a feltétlen és feltételes reflexek tanulmányozására

készült, ugyanakkor képes volt egy zseblámpa fényének követésére, és kondicionálás után egy furulya hangjának irányába is hasonlóan elindult (2.2. ábra).

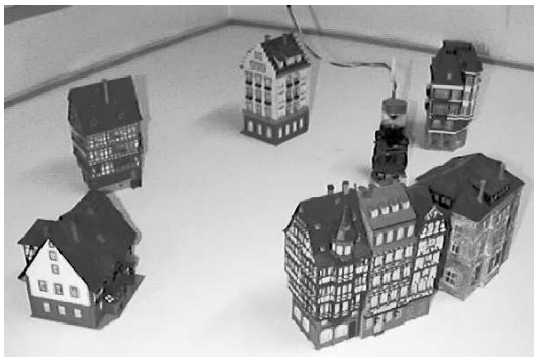
2.2. ábra. A fény és hang követésére képes Szegedi Katica (az Országos Műszaki Műzeum másolata) ([15], Szabó P. engedélyével).

A hetvenes évektől kezdve növekszik az irodalma az előbbinél összetettebb, *helyfelismerésen alapuló tájékozódásnak* ([122]). Ebben alapvető szerepe van a rágcsálóknál előszeretettel tanulmányozott, a hippokampusz principális sejtjeiként azonosított helysejteknek[4], melyek a környezet egy jól meghatározott kis területén válnak aktívvá ([52]), illetve a fejiránysejteknek[5], melyek a fej orientációját adják vissza az aktuális pozíciótól függetlenül. E két funkció segítségével az állat képes bizonyos helyeket és célirányokat megjegyezni, majd később visszatérve az emlékek alapján helyes irányba továbbindulni ([190]). Az eljárás robotok navigációjában való alkalmazására példa többek között Takács és Lőrincz ([179]), valamint Chokshi és társai ([30]) munkája.

Az eddigiekhez képest komoly továbblépést jelent, amikor az élőlény a helyek felismerésén túl képes a helyek közötti kapcsolatot, közvetlen átjárhatóságot is valamilyen módon memorizálni, képes létrehozni a környezet *topológiai térképét*. Az eljárás igen népszerű navigációs módszer mobil robotok esetében, mivel kiemeli a feladatvégzés szempontjából lényeges tulajdonságot, a kitüntetett helyek közötti haladást. A topológiai gráfot használó eljárásokra általában jellemző, hogy a szenzorok körét úgy határozzák meg, hogy azok képesek legyenek bizonyos helyeket egyedileg azonosítani ([58]). Ennek érdekében gyakran az is előfordul, hogy a kutatók egyszerűsített környezeteket használnak vagy a robotok csak

[4] place cell
[5] head direction cell

meghatározott útvonalakon, így például falak mentén közlekedhetnek, ezáltal is segítve a fontos helyek megkülönböztetését ([92], [109]). Topológiai navigációt végző robotot mutat a 2.3. ábra.



2.3. ábra. Navigációs kísérleti terep. A modellházak között egy Khepera robot mozog, melynek tetején egy panorámakamera látható ([58], M. Franz engedélyével).

A tájékozódás még összetettebb szintjét a teljes terep felmérése vagy másképpen a *metrikus navigáció* jelenti, melynek használatát rágcsálók, kutyák, főemlősök esetén lehet igazolni. Egy ilyen jellegű térképen a környezet geometrikus jellemzői, távolságok és irányok tárolódnak, hasonlóan egy kétdimenziós, fölülnézeti térképhez. A metrikus navigációt használó élőlények útvonalösszegzést végezve határozzák meg egyes helyek, tárgyak távolságát, egymáshoz viszonyított irányát ([145]). Ez egy részletgazdagabb reprezentáció, ami lehetővé teszi rövidebb utak kipróbálását ismeretlen részeken át, illetve két hely között a legrövidebb út számítását is ([190]).

A navigáció egy, az eddigiektől kissé eltérő útja a csoportosan élő, kooperatív állatoknál, így hangyáknál, termeszeknél figyelhető meg. Ezek az élőlények is képesek lehetnek helyfelismerésre — és így útvonaluk pontosítására — ételszállítás során ([32]). Ezen azonban túlmutat a feromont használó, *környezeten keresztüli kommunikáció*, aminek segítségével ételforrásokat találnak meg, ismételten visszatérnek hozzá, és eközben egy nagyjából optimális utat alakítanak ki, mely a környezet változásához is képes alkalmazkodni. Ezzel a metrikus jellegű kognitív térképet nem saját magukban, hanem a környezetben hozzák létre ([29]). Eljárásuk egyúttal útoptimalizációs algoritmusok egész családjának ihletője is lett ([50]).

2.3. Robotok navigációja

A robotika kezdeti célkitűzéseiben a tájékozódásnak nem jutott komoly szerep. Az első robotok ipari automataként dolgozó robotkarok voltak, melyek mesterségesen létrehozott, előre meghatározott, rögzített, vagyis jól tervezhető környezetben dolgoztak és dolgoznak ma is. Ezen robotok esetében mindig van egy referenciapont — a rögzítési pont —, amihez képest végzik mozgásukat, így „csupán" mozgástervezésre van szükségük, ami jórészt megoldott feladat. A később fejlődésnek indult mobil robotoknál a feladat annyival nehezebb, hogy egyrészt nincsen referenciapont, másrészt a feladatvégzés környezete is jóval gazdagabb.

A korai mesterséges intelligencia robotokkal kapcsolatos eredményei, így a kockapakolás mikrovilágokban, a gyors, látványos sikerek ellenére sem tudtak igazi áttörést hozni a számítógépen kívüli, nem a programok számára előfeldolgozott, azaz kevésbé absztrakt feladatok megoldásában.

A nyolcvanas évektől kezdve egyre több kutatás célja volt az, hogy a navigáció különböző részfeladatait sikerrel megoldják, nagyjából valós vagy legalábbis kevéssé módosított környezetekben.

Az egyik, korábban már említett, egyszerű reaktív megoldás a Braitenberg-járművek csoportja, melyek kombinálásával akadálykikerülést és érdekes tárgyak megközelítését végző összetett trajektóriákat lehet létrehozni, mindezt környezeti térkép készítése nélkül ([20]).

Valamivel összetettebb viselkedést mutató, de szintén hosszú távú memória nélküli szimulált robotcsótányt hozott létre Beer kis neuronhálózatok serkentőgátló kapcsolatainak egymásra hatásából ([14]). A hatlábú robot falkövetést és táplálékkeresést is képes végezni.

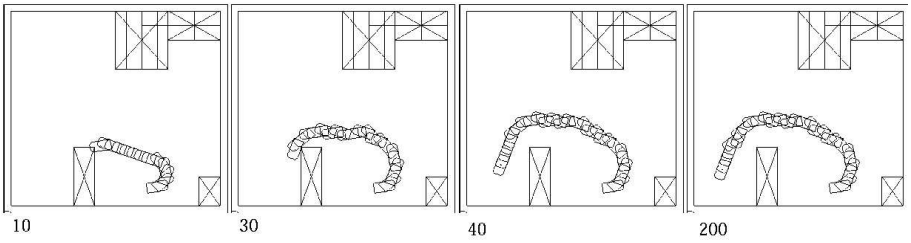
Connell Herbert nevű valódi robotja Brooks viselkedésalapú módszertanát követi, azaz irányítását az elakadásmentes mozgás, az akadálykikerülés, a falkövetés és a stratégia egymásra épített, párhuzamosan működő rétegei oldják meg. A tájékozódásban egy iránytű és a falak segítenek. A palackok megtalálása memória nélkül, véletlenszerűen történik, a robot nem használ térképet. A kiindulópontra visszajutáshoz a robot azzal az alapfeltevéssel él, hogy eredetileg a legdélebbi szobából indult, ezért a begyűjtés után minden ajtón áthaladva dél felé fordul ([34]).

A tájékozódás hatékony megvalósításához az eddigi előre definiált, reaktív viselkedésen túl az *adaptáció* képessége új lehetőségeket jelent. A külvilág modelljének, térképének megalkotása, de legalábbis az adott környezet egyes jelleg-

zetességeihez alkalmazkodás lényegesen javítja a robot teljesítményét, miközben csökkenti a tervezőre jutó terheket ([64]). A következő példák a különféle adaptációs eljárások példái a robotok navigációjának témaköréből.

Wyeth backpropagációs neurális hálózatot használt robotja tanítására ([200]). A kamerával rendelkező ágens a tanító jelzései alapján tanulta meg, hogy a kép melyik szélén található összegyűjtendő labda vagy kikerülendő akadály, és később, a tesztelés szakaszában, hasonló körülmények között 80%-os megbízhatósággal szedte össze a labdákat.

Nehmzow és Smithers Kohonen-típusú önszerveződő neurális hálót használt arra, hogy egy falkövető robot felismerje a különböző sarkokat azok iránya és egymástól való távolsága alapján ([123]). Megfelelő információ esetén — legalább három sarokra visszatekintés — a robot megoldotta feladatát.



2.4. ábra. A robot tanulja környezetét. A Kohonen-térképre alapozott szenzortér bejárhatóságának megtanulása dinamikus programozással ([90], B. Kröse engedélyével).

Kröse és Eecen a Kohonen-hálót egy megerősítéses tanulási módszerrel ötvözték, mellyel szimulált robotjuk sikeresen járta be egyszerű környezetét (2.4. ábra). Az eljárás során a robotot a terep véletlen pontjaira helyezték, amivel az önszerveződő háló a 16 távolságszenzor és az iránytű méréseinek terét kifeszítette ([90]). Mivel a szenzortér szomszédsága nem feltétlenül felel meg a valós térbeli relációknak, sőt az ottani éleket követve a robot ütközhet is, ezért véletlenszerűen generált útvonalak bejárása során egy dinamikus programozási eljárás elkészítette a világmodellt leíró állapotátmenet- és költségfüggvényt ([166]). Bár az eljárás működőképes, valós környezetbe helyezni — az időigényes tanulás miatt — eléggé nehéz.

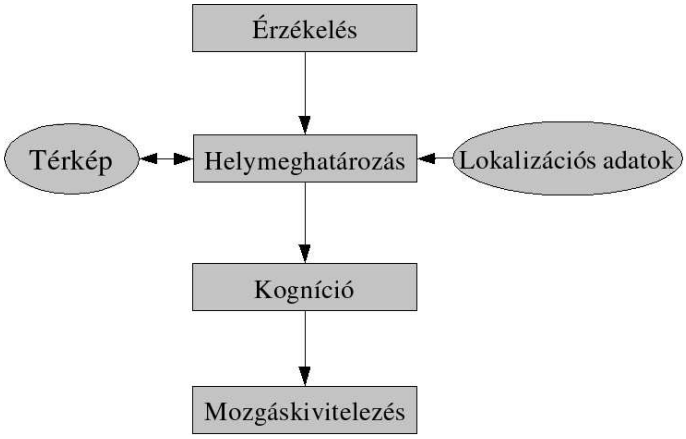
Lee és társai evolúciós programozást használtak egy Khepera optimális irányító moduljainak elkészítéséhez ([96]). A robot feladata egy doboz megközelítése és megfelelő helyre eltolása volt, ami a legjobb modulokból összeállított

egyedeknek sikerült is.

Gyakori megoldás a robothoz tartozó neurális hálózat kialakítása genetikus algoritmus segítségével. Nolfi és Parisi a háló súlyait, míg Kodjabachian és Meyer a háló struktúráját, azaz a neuronok és a közöttük lévő axonok növekedését optimalizálta genetikus eljárással. Előbbinél a robot tárgyak összegyűjtését végzi ([128]), utóbbi esetben egy csótányszerű animat válik képessé a rovaroknál gyakori háromlábú járásmódra[6], akadálykikerülésre és iránykövetésre ([86]).

A kutatásokon túl, a kereskedelmi forgalomban megjelenő autonóm ágensek, például fűnyírók elektromos kerítések között, porszívók mesterséges terelőfalak által határolt térben dolgoznak, miközben a padlón lévő akadályokat érdemes elpakolni, vagyis ezek a megoldások is csupán megszorításokkal képesek elvégezni a tájékozódás feladatát ([172]).

Az itt felsorolt kísérletek jellemző példái a tájékozódás megvalósításának a mesterséges intelligencia módszereinek felhasználásával. Ezek, bár eredményes kutatások, többnyire részmegoldások, melyek a navigáció néhány aspektusát emelik ki, arra próbálnak megoldást adni. Ezzel szemben a jövő robotjainak folyamatosan változó, igen összetett világban kell működniük, egyáltalán nem steril vagy egyszerűsített körülmények között. Ez egyben annyit is jelent, hogy sokkal életszerűbb, emberszerűbb, de legalábbis komplikáltabb világmodellt kell alkotniuk ahhoz, hogy sikeresek legyenek.



2.5. ábra. A tájékozódás lényeges elemei.

A 2.5. ábra a navigáció általánosan elfogadott lényeges elemeit és azok össze-

[6] tripod gait

függéseit jeleníti meg. A robot helyzetének meghatározása az érzetek összességére, a lokalizációs adatokra és a térképre alapozott pozícióbecslés során történik. A bejövő információk alapján pontosított helyzetnek megfelelően módosul a térkép: a pontatlan mérések javulnak, az eddigi helyes mérések megbízhatósága megnő, új információk jelennek meg az eddig még nem érzékelt térrészen, és esetleg korábbi, hibás információk törlődnek. A javított térképet és rajta önnön elhelyezkedését felhasználva — céljai ismeretében — a robot megtervezheti jövőbeni mozgását, melyet a mozgáskivitelezés során válthat valóra. A rendszer összetevőit részletesebben a fejezet hátralevő része mutatja be. A teljes folyamat elején és végén elhelyezkedő feladatok, vagyis az érzékelés és a mozgáskivitelezés kevésbé információelméleti, mint inkább fizikai kérdéseket vetnek föl, úgymint jelterjedés, zaj keletkezése, érzékelési modalitások jellemzői, kinematika, dinamika, manőverezhetőség, emiatt ezekre nem is térek ki bővebben. Ezeket a témákat részletesen tárgyalja Siegwart és Nourbakhsh ([155]).

2.3.1. Helymeghatározás

Elsőre úgy tűnhet, hogy a helymeghatározás egy már megoldott feladat. A robotra kell tenni egy GPS[7] eszközt, mely visszaadja a robot pontos földrajzi koordinátáit a trilateráció módszerét felhasználva ([16]). Ez azonban nem ennyire egyszerű. A rendszer specifikációja nagyjából százméteres eltérést engedélyez a valós értéktől, biztonsági okok miatt. Ezt a hibát a differenciális GPS módszerével egy ismert pozíciójú jeladót felhasználva 4-6 méterre lehet csökkenteni, de ezt tovább rontja, hogy a jel különleges domborzati viszonyok között, masszív épületekben, alagutakban nem mindig áll rendelkezésre, csupán az esetek 90%-95%-ában. Ez nem elegendő a porszívózásnál vagy egy asztalon rendet rakó robotnál, főképp nem a nanorobotok esetében ([26]).

Elképzelhető olyan külső természetes vagy mesterséges jelek felhasználása, melyek állandóan érzékelhetőek, és így biztosítják a pozíció mindenkori becslését. Az előbbire példa a Nap vagy a csillagos ég lehet, ahogy az a hajózásban évezredek óta ismert ([151]). Mesterséges jelek alkalmazása is több évtizedes múltra tekint vissza: automatizált irányított járműveket[8] előszeretettel használnak raktározási, épületen belüli szállítási, idegenvezetési feladatokra. Ezekben az

[7] Global Positioning System – globális helymeghatározó rendszer
[8] AGV – Automated Guided Vehicle

esetekben a padlóra festett csíkok, leragasztott mágnesszalagok, beépített jeladók vagy az ajtók mellett elhelyezett robot által leolvasható vonalkódok orientálnak ([155], [53]), ahogy az a 2.6. ábra példáin is látható.

(a) Mikrofonok a falon. A robot a felső végén lévő jeladóból ultrahangot bocsát ki, amit a fali mikrofonok érzékelnek. A központba beérkezett jelek eltéréséből a robot pozíciója kiszámítható ([195], P. Aarabi engedélyével).

(b) Múzeumi terepmódosítás. A múzeumot bejáró robot automatikus energiatöltéséhez más tárgyakkal nehezen összetéveszthető jelek, úgymint a falon egy rózsaszín négyzet, a padlón fekete-sárga csíkok, segítenek a pontos pozícionálásban. ([129], I. Nourbaksh engedélyével).

2.6. ábra. A környezet módosításának két lehetséges fajtája az egyszerűbb tájékozódás érdekében.

Ezek azonban nem általános érvényű, korlátozottan használható eljárások, ráadásul jelentős többletköltségeket is jelenthetnek a mesterséges megoldások esetében.

A természetes jelek, tereptárgyak használhatók oly módon is, hogy bár nem mindig látható közülük egy kitüntetett, mégis a jelek egy részhalmaza már egyértelmű pozícióbecslést tesz lehetővé. Ha ezek a jelek egyedien azonosíthatók, és mindig érzékelhető belőlük legalább három, akkor triangulációval vagy trilaterációval a robot helyzete meghatározható ([12], [103]).

Ha az aktuálisan érzékelt környezet alapján a robot pozíciója legalább kitüntetett helyeken meghatározható, és ezen helyek között a robot megbízhatóan tud közlekedni, akkor ennyi információ már elég lehet egy térkép létrehozásához és a sikeres működéshez ([58], [130], [102]).

Ha nem áll rendelkezésre olyan, a külvilág ingereit érzékelő eszköz, amivel a robot pozíciója meghatározható, akkor fölmerülhet még az odometria módszere

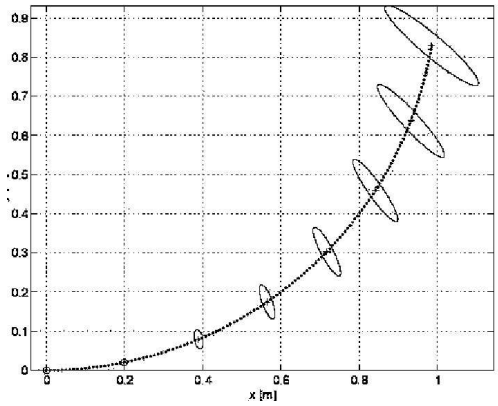
([155]). Ennek során a robot proprioceptív receptorait, azaz lépésszámlálóját vagy kerekek esetén azok elfordulásszámlálóját használja. Ekkor a robot a kiinduló pozícióhoz képest összegzi a változásokat, felhasználva saját paramétereit (lábhossz, kerékátmérő, tengelytáv), amiből az aktuális állapotra következtet. Így például a 2.1. képlet egy síkban mozgó, differenciálművel felszerelt kétkerekű robot odometrikus pozícióbecslésének egy lépését mutatja, ahol x, y és θ az aktuális koordináták és irány, Δs_l és Δs_r a bal és a jobb keréken mért elmozdulás, míg b a kerekeket összekötő tengely hossza.

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} + \begin{bmatrix} \frac{\Delta s_r + \Delta s_l}{2} \cdot \cos(\theta + \frac{\Delta s_r - \Delta s_l}{2b}) \\ \frac{\Delta s_r + \Delta s_l}{2} \cdot \sin(\theta + \frac{\Delta s_r - \Delta s_l}{2b}) \\ \frac{\Delta s_r - \Delta s_l}{b} \end{bmatrix} \tag{2.1}$$

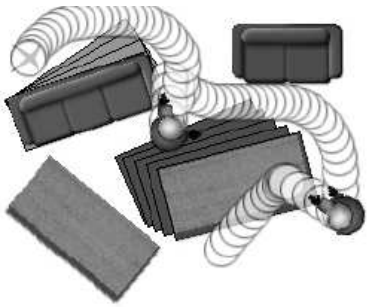
Az odometriához hasonló megközelítés az inerciára alapozott navigáció, melynek során giroszkóp határozza meg a robot aktuális irányát és gyorsulásmérő a sebességváltozását ([16]). Míg az előbbi pontossága meghaladja a mobil robotok navigációjához szükséges szintet, a gyorsulásmérők működése során összegyűjtött hiba miatt az eljárás a gyakorlatban nem alkalmazható.

Bár az odometria és az inercia nem építenek a környezet jellegzetességeire, önmagukban általában mégsem oldják meg a problémát. A cselekvések és a mérések során fellépő zaj halmozódik, ami egy idő után általában használhatatlanná teszi a becslést (2.7. ábra). A szisztematikus zajokat — például kerekek eltérő mérete, szabálytalan alakja, a szenzor véges felbontása, kis mintavételezési frekvencia — bizonyos mértékig lehet ellensúlyozni, de a véletlen zajok, így a padló egyenetlenségei, a lábak megcsúszása vagy az ütközés ellen nincs hatékony védekezés ([17]).

De ha el is tekintünk a felsorolt módszerek használatának nehézségeitől, a helymeghatározás nem csupán a robot koordinátáinak kiszámításáról szól. A robotnak a saját helyzetén túl a vele interakcióba kerülő tárgyakat is ismernie kell (2.8. ábra), a tájékozódását befolyásoló mozgó akadályokat figyelembe kell vennie ([164]). Tudnia kell, hogy hol található az újratöltő állomása, és hogy miként juthat el oda. Ha munkakörnyezetében emberek mozognak, mint ahogy a nem mesterséges körülmények között működő robotoknál ez gyakori lehetőség (2.9. ábra), a robot emberekhez viszonyított relatív pozíciója is meghatározó ([194], [28]).



2.7. ábra. Az odometria hibája egy robot balra kanyarodásakor. A növekvő ellipszis a pozícióbecslés pontatlanságának növekedését jelzi, mely a mozgás irányára merőlegesen nagyobb, mint a mozgás irányában. Az ellipszis elfordulása az orientáció hibabecslésének változását is jelzi ([155], R. Siegwart engedélyével).



2.8. ábra. Egy pakoló robot, mely bútorokat tologat, hogy célhoz érjen ([164], M. Stilman engedélyével).

2.9. ábra. A tömeg a robot körül nehezíti a lokalizációt és a tervezést is ([129], I. Nourbakhsh engedélyével).

Valószínűségi módszerek a lokalizációban

Az érzékelésben és a helymeghatározásban jelentkező nagyfokú bizonytalanság kezelésének előszeretettel alkalmazott módszere valószínűségi alapokon nyugszik. Ezen eljárások két fő logikai lépésből állnak. A cselekvés frissítése során az utoljára meghatározott állapotban (s_{t-1}) bekövetkezett cselekvések hatását tükröző odometrikus információ (o_t) határozza meg a robot új állapotbecslését.

$$s'_t = Act(o_t, s_{t-1})$$

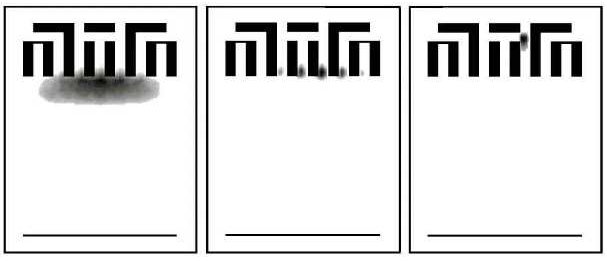
Az érzetek frissítésekor a robot exteroceptív szenzorai által begyűjtött adatok (i_t) pontosítják az előbbi becslést (s'_t) egy új aktuális állapottá (s_t).

$$s_t = See(i_t, s'_t)$$

Az így általánosan megfogalmazott valószínűségi eljárás két leggyakrabban alkalmazott osztálya a Markov-lokalizáció és a Kálmán-szűrőn alapuló lokalizáció.

Az általánosabb Markov-módszer során a robot több, tetszőleges valószínűségi eloszlással meghatározott lehetséges pozíciót tart nyilván (2.10. ábra). A cselekvés frissítése során összegezni kell mindazt a valószínűséget, ahogy a robot az előző lehetséges állapotokból az aktuálisnak vélt állapotba az összes potenciális úton eljuthatott. Az új pozícióeloszlás kiszámításakor pedig az összes állapotra ki kell számolni, hogy az aktuális érzetek és a térkép alapján mi az ott tartózkodás valószínűsége. Az érzékelés frissítése a Bayes-módszer használatával, a térkép és a robot felépítése segítségével számítható.

A módszer Markov-tulajdonságot feltételez, vagyis azt, hogy az aktuális pozíció csak az előző pozíció, a robot mozgásával kapcsolatos érzetek és a külső ingerek függvénye, ami nem mindig igaz. A módszer másik hátránya a nagy memória- és processzorigényben rejlik, hisz minden pozíción minden lépésben számítást kell végezni. Markov-lokalizációt használó tájékozódási eljárást tartalmaz a Dervish nevű robot, mely az 1994-es AAAI robotversenyt megnyerte ([130]) és Rhino is, mely egy bonni múzeumban végzett tárlatvezetést több napon keresztül ([184]).



2.10. ábra. Összetett becslés és pontosítása. A bal oldalon a sötét felhő jelöli a viszonylag pontatlan pozícióbecslést, mely a folyosóba érve több részre esik szét a középső ábrán. A jobb oldali ábrán, az elágazáshoz érve ezt a Markov-lokalizáció újra egyesíteni tudja ([27], W. Burgard engedélyével).

A praktikusabb Kálmán-szűrő korlátozottabb alapfeltételezéssel él. A robot pozícióját mindössze egy Gauss-eloszlással közelíti. Az eljárás odometriával kiszámítja a várható pozíciót. Ezt a térképpel együtt arra használja, hogy a tereptárgyak pozícióját jósolja. Az eredményt összeveti a tényleges érzetekkel és a Kálmán-szűrés végeztével előáll a legvalószínűbb új pozíció.

A módszer előnye, hogy így csupán két paraméterrel, a várható értékkel és a szórással megadható a robot mindenkori hiedelme. Másik haszon, hogy a különböző szenzoroktól származó információkat nagyon könnyű egyesíteni. A szenzorfúzió során az érzékelők önálló Gauss-eloszlású becslései egy közös normális eloszlássá alakulnak. A Kálmán-szűrő hátránya, hogy a normális eloszlású zaj feltételezése nem teljesen realisztikus, illetve, ha a robot a hibák felgyülemlése miatt eltéved (ez úgy jelentkezik, hogy a szórás komolyan megnő), akkor — az egyetlen pozíció kezelése miatt — nem tud más feltételezésre váltani (korrespondancia-probléma). Kálmán-szűrő használatára példa Smith és társai ([162]), valamint Leonard és Durrant-Whyte munkája ([98]).

2.3.2. A térkép

A helymeghatározás folyamatának az érzékelés melletti fő bemenete a térkép. Jóval egyszerűbb az ágens feladata, ha a térkép eleve adott, de gyakran nem ez a helyzet. A térkép létrehozása lényegesen könnyebb, ha a külvilág absztrakt módon, névvel ellátott objektumok formájában érzékelhető, mint ahogy az a robotfutball szimulátorában is jellemző ([170]). Ez azonban a legritkább esetben van így, a robothoz eljutó érzetekből — így például ultrahangok visszaverődése vagy egy kétdimenziós, színes kép percepciója — meglehetősen indirekt módon következik a külvilág felépítése, a két rész közötti távolságot a robotnak kell áthidalnia, amint arra Siegwart és Nourbakhsh rámutat ([155]).

Valódi térképkészítésre és -használatra már a legkorábbi mobil robotok esetében is volt példa. A Stanford Cart volt az egyik első összetett világmodellt alkalmazó robot, mely teljes háromdimenziós környezeti térképet hozott létre sztereó látás segítségével ([118]). Azonban a modell felépítése akkora számításigénnyel járt, hogy a robot húsz métert nagyjából öt óra alatt volt képes megtenni.

A térképezéssel kapcsolatos másik végletet Brooks képviseli, aki munkáiban amellett érvel, részben a Stanford Carttal tapasztaltak miatt is, hogy intelligencia létrehozható reprezentáció és központi irányítás nélkül is. Ezért az általa készített

robotok nem tartalmaznak teljes navigációs rendszert abban az értelemben, hogy céljuk „csupán” egy adatvezérelt, a külvilág által irányított mozgás a térben ([24]). Azonban a tér alapos felmérése, a külvilág reprezentációja nélkül optimális tervek létrehozása, hosszú távú döntések meghozatala nehezen kivitelezhető, ezért a tájékozódást középpontba helyező kutatások szinte mindegyike használ térképet.

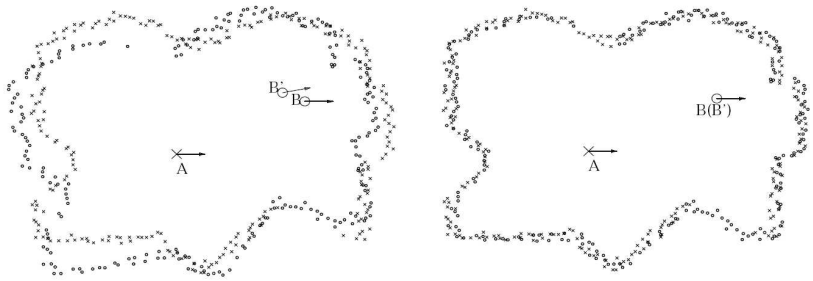
A térkép lényeges tulajdonsága, hogy milyen módon ábrázolja a környezetet. Sokféle reprezentáció képzelhető el, ami a rendszer többi részének választását is befolyásolja. A térkép készítése mindig ábrázolandó elemekre bontással és absztrakcióval jár. Ez egyúttal információvesztést is jelent, aminek célja az, hogy a tájékozódás szempontjából igazán fontos jellegzetességek kiemelődjenek, az irreleváns részek viszont elhagyhatók. Fontos, hogy a térkép felépítésének és felbontásának összhangban kell állnia az érzékelők adta lehetőségekkel és a feladat megkívánta részletességgel. Ezen kívül a választás függ attól is, hogy a térképet készen kapja a robot, vagy saját magának kell előállítania méréseiből.

A térképi reprezentáció egyik legkézenfekvőbb módja poligonok ábrázolása a globális koordinátarendszerben. A módszer előnye, hogy teljesen pontos térkép készíthető, hátránya viszont a jelentős számításigény. Az utóbbi javítása érdekében végzett egyszerűsítés lehet, hogy csupán egyes határoló vonalakat ábrázol a robot, melyeket a távolságérzékelők pontszerű mérései alapján rak össze valamilyen egyenes-, illetve görbeközelítő eljárással. Az így születő térképek a jellemzőalapú[9] reprezentáció kategóriájába tartoznak. A robot kiválasztja a számára érzékelhető, fontos tereptárgyakat, majd ezek absztrakt leképeződésének összessége lesz a térkép ([94], [98]).

A térkép ábrázolásaként a Lu és Milios által bevezetett direkt módszer is alkalmazható ([102]). Ilyenkor a robot az érzeteket, így például a távolságmérések kétdimenziós eredményeit, előfeldolgozás nélkül tárolja, és a pozícióbecslést az aktuális érzetek és korábbi mérések összehasonlításával kapja meg (2.11. ábra). Az eljárás hátránya a nehézkes bizonytalanságkezelés, illetve ha eltéved a robot, nincs hová visszatérnie.

Igen gyakori módszer a diszkrét reprezentáció, amikor valamilyen dekompozíciós eljárás során a terep kisebb részekre darabolódik. A kisebb elemek — általában a külvilág egyéb jellemzőit figyelmen kívül hagyva — arról tárolnak információt, hogy az adott terület foglalt vagy bejárható-e. A módszer különféle

[9] feature-based

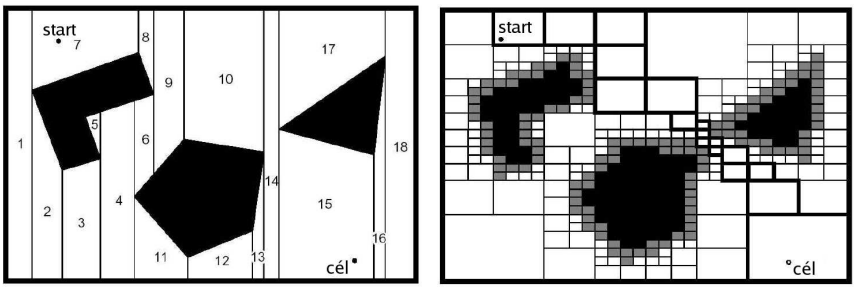


(a) Illesztés előtt (b) Illesztés után

2.11. ábra. Direkt reprezentáció ([102], E. Milios engedélyével)

változatai a részekre vágás eltérő algoritmusait használják.

Az egzakt celladekompozíció esetén a felbontás az érzékelt akadályoktól függ, és eltérő formájú cellákat eredményez. Az üres részek kontúrja a tárgyak körvonalát követi. A határoló vonalak az akadályok csúcsain átmenő élekből keletkeznek, konvex formákat hozva létre. Az eljárás azzal az alapfeltételezéssel él, hogy egy ilyen területen belül nem fontos a robot pozíciója, miközben a térbeli szomszédságok jól definiálhatók. A módszer hátránya, hogy mérések alapján, különösen sok objektum esetén, nehezen állítható elő ([161]).



(a) Egzakt celladekompozíció (b) Adaptív celladekompozíció

2.12. ábra. Dekompozíciós stratégiák azonos környezetben alkalmazva ([155], R. Siegwart engedélyével).

A fix celladekompozíciónál nem kell a robotnak a határoló éleket a mérések alapján kijelölnie, mert egy előre meghatározott négyzetháló feszül ki a terület fölött, amelynek egyes cellái foglaltak, mások szabadok. A módszer rendkívül nép-

szerű alesete a foglaltsági háló, melynél a cellák nem binárisak, hanem foglaltsági valószínűségeket tárolnak. Néhány példa erre a reprezentációra Elfes ([51]), Martin és Moravec ([106]), valamint Thrun ([182]) munkája. Így a mérésekben rejlő bizonytalanság könnyebben kezelhető, és a reprezentáció Markov-lokalizációhoz is használható. Megfelelő frissítési szabály felhasználásával elérhető, hogy a térkép a korábbi eredményeit az aktuális értékekkel integrálja. Foglaltsági hálóval a következő fejezetek részletesebben is foglalkoznak. A fix dekompozíció hátránya, hogy a lefedettségtől függetlenül nagy a memóriaigénye, és egy mesterséges léptékű és irányú felbontást ad, ami nincs feltétlenül összhangban a feladattal. Ezen kívül, a véges felbontás miatt, a foglalt cellák összeérésével bizonyos utakat nem vesz számításba az útvonaltervezés.

Kompromisszumos megoldás lehet az adaptív celladekompozíció, amely a foglaltságtól függő cellaméreteket eredményez. A rekurzív felbontás négyfelé vág minden cellát olyan mélységig, amíg egy cella vagy teljesen üres vagy teljesen foglalt nem lesz. Ez a módszer egyrészt memóriatakarékosabb, másrészt alkalmazkodik a környezet bonyolultságához, ugyanakkor bonyolultabb számítások szükségesek hozzá ([93]).

Egy következő reprezentációs kategória a topológiai, mely gyakran az előző térképábrázolási módszerek valamelyikére épül, vagy legalábbis tartalmaz metrikus elemeket. Topológiai térkép használatakor a robot az eddigi megközelítésekkel szemben nem a terep geometriai jellemzőire fókuszál, hanem kitüntetett helyekkel és közöttük lévő kapcsolatokkal dolgozik, vagyis egy alapvetően metrikus térkép helyett többnyire a környezet topológiai gráfját állítja elő ([130], [58], [182]). Ez a megoldás útvonaltervezéshez jobban alkalmazható, és az előbbieknél általában kisebb felbontású reprezentációt jelent, ami ebből kifolyólag kisebb memóriaigényű is. Ugyanakkor a kompaktabb tárolás miatt olyan részletek is elveszhetnek, ami egy nagyjából üres, információszegény térrészen a robot helymeghatározását segítené, és az eltévedés kockázatát csökkentené.

2.3.3. Útvonaltervezés

Ha az ágens valamilyen módon tudja a saját helyzetét és ismeri a környezetét, akkor végre tényleg mobillá válhat, aktuális pozíciójából a cél felé indulhat. Annak meghatározása, hogy ez miként történjen, egy kogníciós folyamat során zajlik le. Ennek két lényeges összetevője az útvonaltervezés és az akadálykerülés, a feladat

stratégiai és taktikai szintje.

Érdekes módon az útvonaltervezés problémája először a helyhez kötött gépeknél, a robotkaroknál jelentkezett. Mivel ezek a manipulátorok általában igen összetett, hat szabadságfokkal rendelkező eszközök[10], ezért vezérlésük megfelelő tervezést igényel. Ráadásul a napi munkavégzéshez talált megoldásnak optimálisnak kell lennie az idő és a költség függvényében is. Ehhez képest két dimenzióban közlekedő, többnyire kereken guruló robotok útvonalának megtervezése valamivel egyszerűbb feladat.

A rögzített robotok útvonalának megtervezésében általánossá vált a robot konfigurációs terében elvégezni a számításokat ([150]). Ez az összes paraméter által meghatározott sokdimenziós tér, ennek egy része azon konfigurációk halmaza, melyek beállításakor a robot akadályba ütközne. A konfigurációs térben való mozgás egymásba alakítható állapotokon keresztül történik, így egy megtalált legrövidebb út ténylegesen kivitelezhető.

Bár a mobil robotok esetében a mozgások általában megszorításoknak engedelmeskednek, a kutatók mégis gyakran tekintik robotjukat holonomikusnak[11], ezzel lényegesen egyszerűsítve a problémát. Másik gyakran alkalmazott könnyítés a robot pontszerűvé tétele, miközben a tárgyak kontúrja a robot méretével egyezően növekszik.

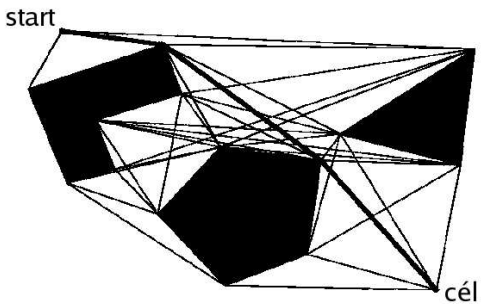
Ezen könnyítések után a három fő útvonaltervezési kategória, az útvonaltérkép, a dekompozíció vagy a potenciamező valamelyikét szokták alkalmazni ([155]).

Az első típusnál az üres terek lehetséges útvonalainak hálózatát kell létrehozni, amit a robot követhet. Ez lényegesen eltérő módokon is megvalósulhat. A láthatóságigráf-módszer az akadályokat mint poligonokat kezeli és, éleket húz be az összes egymást „látó” csúcs, valamint a kiinduló és a végpont közé ([134]). A legrövidebb út az élek mentén alakul ki (2.13. ábra). Az eljárás előnye, hogy optimális utat hoz létre, ugyanakkor azt a veszélyt hordozza magában, hogy a csúcsban elakad a robot. Továbbá nagyszámú, nem teljesen szabályos objektum esetén a gráf nagyon elbonyolódik.

A másik lehetséges végletet a Voronoi-diagram képviseli, azon pontok összességeként, melyek egyenlő távolságra vannak két vagy több objektumtól. Az így

[10] Vagyis a tér bármely pontjára, bármilyen pozícióban eljuthat a munkavégző fej.

[11] Egyszerűen fogalmazva a holonomikus robot annyi függetlenül állítható paraméterrel rendelkezik, amennyi paramétere van.



2.13. ábra. Útvonaltervezés láthatósági gráffal. A vastag vonal a tervezett út ([155], R. Siegwart engedélyével).

meghatározott élek mentén haladva a robot ugyan távolról sem optimális utat jár be, és üres térrészre keveredve eltévedhet, ugyanakkor az elakadás veszélyét csökkenti. Ráadásul ez a struktúra mozgáskivitelezésre is használható, a definíció alapján a szenzorértéket a robot két oldalán egyező szinten tartva a robot a diagramon haladhat célja felé ([31]). Útvonaltérkép létrehozásáról az ötödik fejezetben írok bővebben.

A dekompozíciós tervezés esetén az előző alfejezetben bemutatott foglaltsági térképek használhatók összes előnyükkel és hátrányukkal. A környezetet foglalt és szabad területekre osztja a robot, és az üres térrészen keres útvonalat a start és a cél között. Egzakt celladekompozíció esetén a részek szomszédsága alapján megadott kapcsolati gráfon tervez a robot. Ritka térben ez lényeges egyszerűsítést jelent, de nagy sűrűség esetén nem előnyös. Fix dekompozíciónál az útvonalat például a bozóttűz-algoritmus alapján lehet kiszámítani ([11]). Az eljárás minden cella távolságát meghatározza a célpont felől kiindulva, a kezdőpontból pedig ezután minimumot kell keresni. A módszer hátránya a sok cella miatt jelentkező számításigény, és hogy a reprezentáció felbontásából adódóan bizonyos utak elvesznek, velük nem fog számolni az eljárás. Dekompozíciós útvonaltervezést mutatok be a negyedik fejezetben.

A harmadik útvonaltervezési csoportnál a cél vonzó, az objektumok taszító potenciamezőként viselkednek. Az erők eredőjeként előáll egy felület, amin — megfelelő paraméterezés esetén — golyóként gurulhat a cél felé a robot. Az eljárásnak a konkáv objektumok és a lokális minimumok felbukkanása okoz nehézséget, másrészről viszont hasznos tulajdonság, hogy a mezők alapján a mozgáskivitelezés feladata is megoldott ([93]). A Khatib és Chatila által használt kiterjesz-

tett potenciamező két új összetevő bevezetésével optimálisabb, kevesebb fordulást tartalmazó útvonalat kalkulál. Egyrészről a taszítóerők számításánál nemcsak a tárgyak távolságát, hanem a robot hozzájuk viszonyított irányát is figyelembe veszi. Másrészről a robot aktuális sebessége alapján kizárja azokat az akadályokat, amelyek rövid távon nem fogják akadályozni a robotot mozgásában ([84]).

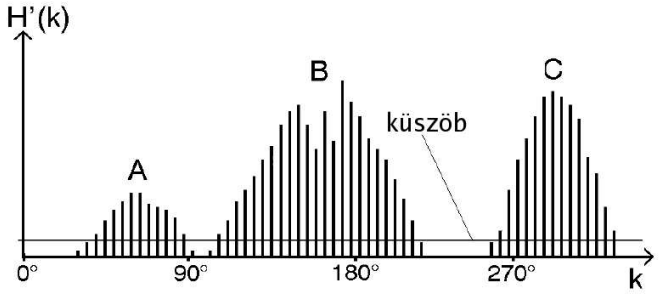
2.3.4. Akadálykikerülés

A navigáció kognitív folyamatának másik, taktikai összetevője az akadálykikerülés. Ennek célja a váratlan akadályok megkerülése oly módon, hogy a robot eredetileg kitűzött célja felé folytathassa útját. Erre elvileg nincsen semmi szükség, hiszen a robot éppen a környező objektumok figyelembevételével tervezte meg útvonalát a kitűzött célig. Mégis adódhatnak olyan esetek, amikor szükség van akadálykikerülésre. Ha a felépített világmodell, térkép pontatlan, akkor a robot ott is akadályba ütközhet, ahol nem számított rá. Változó világban ez pontos térképkészítésnél is előfordulhat. Az is elképzelhető, hogy a robot ismeretlen területen kénytelen áthaladni, s az ott megtalálható objektumok felbukkanására természetesen nem tud előre készülni.

Bár rengeteg algoritmust fejlesztettek ki a kutatók a probléma kezelésére, ebből csak néhány fontosabb típust mutatok be.

Az egyik legelemibb akadálykikerülő eljárás a Bug-algoritmus, melynek több változata is ismert ([81]). A módszer lényege, hogy a robot egyenesen halad a cél felé, majd ha akadályt érzékel, akkor valamilyen irányból addig halad mellette fix távot tartva távolságérzékelője segítségével, amíg a cél iránya ismét szabaddá nem válik. Az eljárás — egyszerűsége ellenére is — sok esetben jól működhet, de mivel csak az aktuális érzeteket használja fel, ezért a teljesítmény erősen környezetfüggő.

A vektormező-hisztogram típusú akadálykikerülő algoritmusok családja éppen ezt a gyengeséget igyekszik kiküszöbölni ([18]). A Borenstein és Koren által bevezetett eljárás használatakor a robot egy lokális foglaltsági hálót készít, amibe a korábbi mérések eredményei is integrálhatók. A foglaltsági hálóból egy hisztogram készül, ami a robothoz képest adott irányban foglalt cellák számát mutatja (2.14. ábra). Ezek után azokkal az irányokkal foglalkozik a robot, ahol elég hely van a biztonságos áthaladáshoz. Ezen irányok közül azt választja, amely az aktuális orientáció és a cél helye szempontjából ideális.



2.14. ábra. Egy jellemző vektormező-hisztogram. A B-vel és C-vel jelölt tárgyak komoly akadályok, míg az A-val jelölt egy kisebb, esetleg távolabbi. A 0 és a 250 fok irányában érdemes továbbmenni ([155], R. Siegwart engedélyével).

A buborékfüzér[12]-technika a robotot övező üres térrész egy olyan maximális szeletét definiálja buborékként, melyben a robot akadálymentesen közlekedhet figyelembe véve a robot felépítéséből adódó kinematikus megszorításokat. A célhoz vezető út ismeretében az előre kialakított buborékfüzér belsejében haladhat a robot. Az eljárás inkább útvonaloptimalizálás, mint akadálykikerülés, hiszen a buborékok elhelyezéséhez az utat előre ismerni kell ([139]).

A Simmons által kidolgozott kanyarodásisebesség-technika a robot haladási és fordulási sebességének terében dolgozik ([156]). A tér elérhető pontjait első közelítésben a robot sebességi és gyorsulási paraméterei határozzák meg. Ezt finomítja az akadályok megjelenése a térben. A tárgyak távolsága és azok elérhetősége konstans sebességű ívek mentén a sebességtér újabb részeit zárják ki. A valós időben működő számításokhoz az akadályokat körökkel közelíti a robot. Ezután a fennmaradó irányokból lehet választani a cél ismeretében. Bár a robot kinematikájának és dinamikájának részleges felhasználása hasznos, az akadályok körökké egyszerűsítése gyakran nem megfelelő megoldást eredményez.

A dinamikus ablak elnevezésű megközelítés szintén haladási és szögsebességgel dolgozik. Az eljárás Fox és társai által készített lokális változata egy ablakot jelöl ki a két sebesség terében, melyet a robot a következő időszeletben el tud érni, figyelembe véve a kinematikai megszorításokat. Az eljárás ebből az ablakból zárja ki azokat a helyeket, melyeknél egy akadállyal való ütközés elkerülhetetlenné válna. A többi térrészből a robot orientációja és a cél közötti szögeltérés, valamint az aktuális sebesség alapján választ a robot ([56]).

[12]bubble band

3

Az Artificial Life Creators robotszimulációs verseny

3.1. Versenyek a kutatásban és a robotikában

Versenyezni jó. Bár elsőre meglepőnek tűnhet, de nincs ez másképp a tudomány esetében sem: a versenyzésnek pozitív szerepe lehet a kutatásokra. A verseny jótékony hatása több okra vezethető vissza. Közismert, hogy a megmérettetés önmagában is nagyobb teljesítményre ösztönöz. Ezen kívül ez az eredmények (algoritmusok, ágensek, robotrendszerek) összevetésének lehet egy új módszere, mérőszáma. Szerencsés esetben a verseny végeredménye is hasznos lehet, hozhat olyan megfigyeléseket, melyek utóbb kiemelkedő szerepűnek bizonyulnak, s a korábbi teóriák pontosítását, esetleg átfogalmazását teszik szükségessé.

Ilyen kitüntetett jelentősége van a kétszemélyes játékok területén az iterált fogolydilemma vizsgálatának, hiszen az Axelrod által kiírt verseny is azt a meglepő eredményt hozta, hogy egy rendkívül egyszerű, alapvetően a kooperációra épülő Tit for tat néven híressé vált algoritmus szerepelt a legsikeresebben, ezzel átértékelve a többrésztvevős feladatokban a rendszer által ki nem kényszerített kooperáció jelentőségét az egoizmussal szemben[1] ([5]).

A versenyzés szintén alapvető eleme a mesterséges élettel foglalkozó kutatá-

[1] Axelrod későbbi könyvében leírja, hogy az első világháború lövészárokharcaiban kialakult az a szokás, hogy a szembenálló felek jól meghatározott időbeosztás szerint lövik a másik oldalt. Ezzel egyrészt eleget tesznek a gyakori ágyúzás parancsának, ugyanakkor az ellenség felkészülhet a támadásra, cserébe az ellenfél ágyúzása is hasonlóan megszokható időközökben történik ([6]).

sok tekintélyes részének, még ha nem is feltétlenül különböző kutatók által létrehozott ágensek mérkőznek meg egymással. Ezen kísérletek egyik első és talán leghíresebb képviselője a Tierra ([142], [143]), ahol is virtuális számítógép belsejébe helyezett önreprodukáló programok küzdenek meg a processzoridőért és a memóriáért. A kísérletek számos érdekes eredményt hoztak: az eredetileg tervezett programlények kódja optimalizálódott, ezen kívül meghúzódó semmitevők, paraziták, sőt hiperparaziták alakultak ki, azaz a versenyzés nem várt új jelenségek felfedezéséhez vezetett.

A robotika területén a versenyzés különösen népszerű, már-már sportszerű méreteket ölt, ugyanakkor többnyire tudományos jellegét sem veszíti el ([9]). Az ilyen események a viszonylag olcsó, kísérletezésre alkalmas robotok egyetemi elterjedésével jelentek meg a kilencvenes években. Az első rendezvények az American Association for Artificial Intelligence nevéhez fűződnek. Az 1994-es konferencián már többféle feladat megoldására lehetett benevezni, így szemétgyűjtés végrehajtására több robottal ([7]), illetve irodai környezet bejárására és térképének létrehozására ([88], [56]).

A versenyzés új korszaka a robotfutball[2] megjelenésével köszöntött be, mely a kezdeti néhány lelkes csapatból ([85]) mára két- és háromdimenziós szimulációs, kicsi, közepes, négylábú és humanoid ligákba tömörülő több száz csapat grandiózus eseményévé vált ([177]). Ez a versenyzési forma rendkívül sok kísérletező erőt vonzott magához, ami a témába vágó publikációk nagy számán is tapasztalható, néhány példa [165], [100] és [152]. Szinte teljes újságokat tölt meg a robotfutballal kapcsolatos cikkek sora, melyek — az alapvető viselkedéselemek létrehozásától kezdve, az ágensek közötti kommunikáción át a csapatkoordinációig — mindenféle témakörrel foglalkoznak. Ezen kívül a robotfutball a robotika oktatásában is elismert helyet vívott ki magának ([160]).

A közvetlen hasznot hozó robotok versenyzése az 1995-ös, komoly anyagi károkkal járó kobei földrengés után jelent meg, először a mentésben részt vevő robotok mérkőzésével ([87], [1]), majd folytatódott a tűzoltórobotokkal ([2]), 2006-ra pedig a háztartási robotok versenyzési szabályait dolgozzák ki éppen ([147]).

Az utóbbi időben a technika és az alkalmazott módszerek fejlődése már azt is lehetővé teszi, hogy valódi környezetben, azaz nem modellvilágokban küzdjenek meg egymással a robotok, így kerülhet sor autonóm tengeralattjárók vagy légijár-

[2] http://www.robocup.org

művek ([44]) képességeinek összemérésére. A szabadban végzett kísérletek talán legismertebb példája a Great Robot Race, melynek során autonóm járműveknek kell a szélsőséges feltételeiről ismert Mojave-sivatagban megtenniük 132 mérföldet. A 2004-es versenyen még igen távol álltak a kutatók a feladat teljesítésétől, de 2005-ben a legjobbak már a szintidőn belül értek célba[3] ([116]).

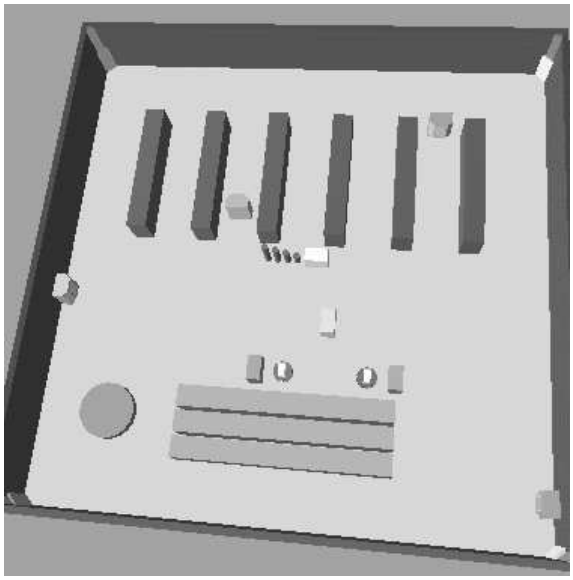
3.2. A verseny kiírása

A Cyberbotics[4] nevű svájci cég, mely a lausanne-i Swiss Federal Institute of Technology egyetem MicroComputing and Interface laboratóriumának együttműködő partnere, az elmúlt években az általuk készített Webots szimulációs programra alapozva számos nemzetközi szimulációs verseny szervezésére vállalkozott ([113]). A lassan hagyományosan másfél évente megrendezett versenyek egyre több versenyzőt vonzanak a különböző témakörökben. Míg korábban a navigáció volt a vállalkozó szelleműek fő feladata, addig újabban dzsúdóversenyek kapcsán mozgáskoordinációban kell kiemelkedőt nyújtani.

Az 1999-ben és a 2000-ben zajló első és második Artificial Life Creators Contest nevű versenyre saját robotirányító programokkal neveztem. A versenyzés célja az alábbi módon foglalható össze. A tárgyakkal véletlenszerű módon betöltött, két négyzetméteres környezetbe helyezett két szimulált 1-es típusú Khepera robot[5] páronkénti mérkőzései során az a versenyző vesztett, amelyiknek előbb fogyott el az energiája. A robotok energiaszintje az idővel egyenletesen csökkent, függetlenül attól, hogy a robot milyen sebességgel mozgott, vagy esetleg állt. Az energiát a terepen elhelyezett négy energiaforrás-modulból lehetett pótolni. Egy feltöltés után a forrás inaktívvá vált, ezért egy energiatöltő előtti „letáborozás" nem jelentett megoldást. A források újratöltődése egyre lassabb lett, vagyis bekövetkezett egy olyan pillanat, amikor már csak egy robotnak elegendő energia állt rendelkezésre. A környezetnek a futások közötti változása miatt nem lehetett előre elkészített útvonalat tervezni az energiaforrások felhasználására. A verseny egy jellemző terepét a 3.1. ábra mutatja.

A vázolt feladat sokban hasonlít a kolibri (*Selasphorus rufus*) nektárgyűjtő viselkedésére ([70]). Ezek a madarak, azon kívül hogy bejárják a környékükön

[3] The DARPA Grand Challenge - http://www.darpa.mil/grandchallenge/
[4] http://www.cyberbotics.com
[5] http://www.k-team.com



3.1. ábra. Az Artificial Life Creators Contest verseny terepe. A sötét falak által határolt terep közepe táján elhelyezkedő kör alakú, fehér tetejű tárgyak a versenyzők. A trapéz felülnézetű szürke „dobozok” az energiaforrások. A többi tárgy a mozgást akadályozza, ugyanakkor a navigációban használható.

fellelhető nektárt adó virágokat, képesek alkalmazkodni azok nektártermelési sebességéhez. A kísérletekből az derül ki, hogy a nap folyamán a gyorsan megtelő virágokat lényegesen többször látogatták, ezáltal a környék erőforrásait csaknem optimális módon használták ki.

A verseny további tulajdonságai voltak lényegesek a tervezés szempontjából.

Az érzékelt környezet a robot számára nyolc taktilis szenzorának észleleteiből, a kereken mért elfordulásból és a kétdimenziós, 80x60 méretű, színes képet szolgáltató kamera által visszaadott képből áll.

A robot nem rendelkezik aktuális pozíciójának valamilyen külső ponthoz viszonyított ismeretével, azaz allocentrikus koordinátáival. Továbbá a pozíció meghatározásához nem lehet felhasználni fix helyű, állandóan látható tárgyat, mivel ilyen nincs.

A környezet zajos, vagyis a távolságmérésre használt infravörös szenzorok, a kerékelfordulás-mérők és a mozgási parancsok eredményében 5-10%-os véletlen zaj jelentkezik.

A tárgyakkal való ütközés eredménye előre nem határozható meg pontosan, de az eredmény függ a robot haladási irányától és az ütközés szögétől.

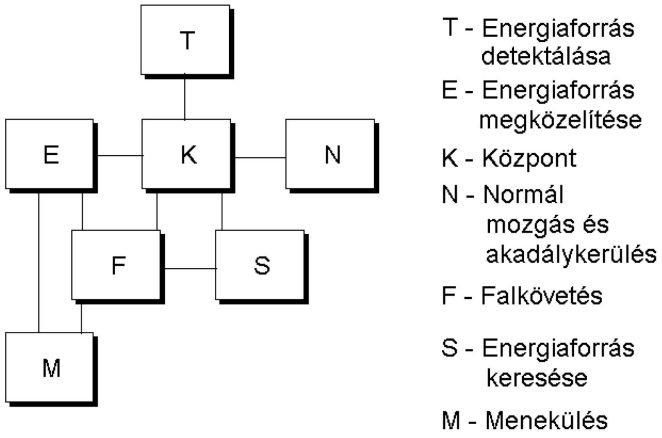
A robotok nem fix időszeletekre kapják meg a vezérlést. A környezet állítja elő az érzeteket, meghívja a robot működését meghatározó algoritmust, majd a válasz alapján beállítja a motor értékeit, és kalkulálja az elmozdulásokat. Azonban a vezérlés annál sűrűbben jut a robothoz, minél gyorsabban fut le az új lépést meghatározó algoritmus. A számításigényesebb kontrollereknél ez azt eredményezi, hogy az utoljára beállított motorértékeknek megfelelően a robot követi addigi trajektóriáját.

3.3. A versenyző

A szimulált robot megtervezésekor legfontosabbnak tartott szempont a hatékonyság volt. A szimuláció számításigényes kontrollereket sújtó időosztásos tulajdonsága arra kényszeríti a fejlesztőket, hogy a mesterséges intelligencia eredetileg alkalmazott időfüggetlen problémamegoldásával szemben a szimuláció futási idejének múlását figyelembe vevő, a környezet jelzéseire aktívan reagáló megoldásokat dolgozzanak ki. Erre azért van szükség, mert a környezet modelljének időigényes előállításával és bonyolult tervgenerálással könnyen kialakulhat az a helyzet, hogy a külvilág tervezés közbeni megváltozása érvénytelenné teszi a tervet, ahogy arra Wooldridge is rámutat. ([198]). Hozzá hasonlóan Brooksot az akkurátusan tervező Stanford Carttal kapcsolatos tapasztalatai a teljesen reaktív, magába foglaló architektúra[6] kidolgozására késztették, mely nem készít világmodellt, nincs valódi belső állapota, a környezet ingereire közvetlenül reagál ([26]). A tervezés és a belső állapot megtartásával is lehet valós idejű eljárásokat alkalmazni, az *anytime* algoritmusokat használva a számítás tetszőleges ponton megállítható és a válasz iteratívan finomodó lehet. Egy másik megoldás több, különféle költségű és hatékonyságú eljárás egyesítése, melyek közül a rendelkezésre álló idő függvényében kell választani, és így a lehető legjobb tervet lehet létrehozni ([60]).

A vázolt elvek figyelembevételével a versenyekre készítettem két hasonló felépítésű, alacsony válaszidejű, reaktív, moduláris felépítésű robotmodellt ([169]), melyek részeit aktivitáson alapuló dekompozíció segítségével hoztam létre ([24]). Ez a hatékony és egyszerű struktúra összhangban van azzal, hogy a szimulált környezet eléggé egyszerű, zajokkal nem túlságosan terhelt. A megközelítés hátrányai: a fáradságos finomhangolási folyamat és — egy bizonyos szint után — a

[6] subsumption architecture



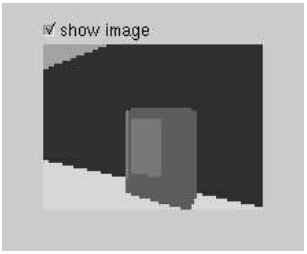
3.2. ábra. A versenyző modulszerkezete. A kamera képére alapozott energiaforrás érzékelése és a többi modul közötti összhangot megteremtő központon kívül a többi modul különféle mozgásmintázatok megjelenéséért felelős.

komplexitás áttekinthetetlen növekedése. A robot moduljainak egymás közötti kapcsolataira világít rá a 3.2. ábra.

A részek közötti összhangot egy központi modul biztosítja, mely az aktuális érzetek begyűjtése után eldönti, hogy melyik modul legyen aktív. A döntést több dolog is befolyásolja. Bizonyos ingerek (az ételforrás megjelenése, a fal közelsége) a megfelelő modult közvetlen reakcióra késztetik. Ha ilyen kiemelt jelentőségű inger nincsen, akkor a robot folytathatja korábbi tevékenységét, hisz némelyik az időosztás miatt csak a szimuláció több iterációján keresztül végezhető el. Ezen kívül az energiaéhség mint motiváció is szerepet játszik a végrehajtandó viselkedés kiválasztásában. Ha a robot elég régen nem találkozott energiaforrással, s ebből következően az energiaszintje lecsökkent, akkor adott környezetben a körülnézés és a falkövetés moduljai aktiválódnak.

Bár a központi modul alkalmazása eltér a Brooks által leírt magába foglaló architektúra azon jellegzetességétől, hogy a modulok egymás közötti interakcióiból, centrális vezérlés nélkül alakul ki az összetett viselkedés, mégis a közvetlen reagálás a külvilág ingereire, a modulok folytatásra való igénye és a motivációk együttesen egyfajta modulok közötti versengést okoznak, ahogy az Brooks rendszerében is megfigyelhető ([22]). A motivációk használata önmagában is a robot animatszerűségét hangsúlyozza, és bizonyos mértékben hasonlóvá teszi a kontrollert Maes viselkedésalapú rendszeréhez ([104]).

Mivel a robot feladata a folyamatos energiatöltés, ezért a többi modul feladata



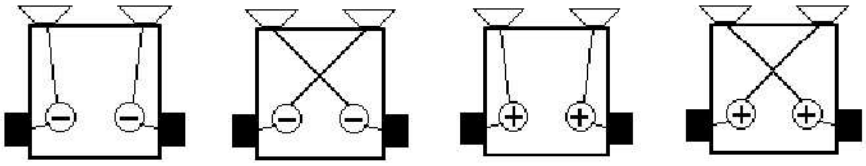
3.3. ábra. Egy energiaforrás látványa a robot nézőpontjából, 45 fokos szögben megközelítve.

3.4. ábra. Az előfeldolgozott kép, mely az erőforrások kímélése miatt csupán a kamera képének egy olyan sorát emeli ki, amelyet az ellenfél nem zavarhat meg, és még beleesik a képmezőben lévő forrás.

az elakadás elkerülése és az energiaforrások megtalálása. Ez utóbbi megvalósításához az infravörös érzékelőket csak egészen közelről lehet használni, míg a kamera nagyobb távolságban is használható, ezért erre a célra a vizuális szenzort választottam. A 80x60-as méretű kép egy nagyságrenddel összetettebb információforrás, mint a nyolc taktilis érzékelő, ezért úgy tűnt, hogy a hatékony működéshez elengedhetetlen a kép előfeldolgozása. Erre a célra egy külön modul szolgál, mely a színes kamera szemmagasságú sorában lévő képpontokról kellően hibatűrő módon eldönti, hogy energiaforráshoz tartoznak-e vagy sem. A robusztusságra a fény–árnyék hatások miatt van szükség, árnyékban az energiaforrás is sötétebb. A 3.3. és a 3.4. ábra a robot által látott képet és az előfeldolgozás utáni állapotot mutatja be.

A kép nem véletlenül csak egy képsorra és csak az energiaforrás színére van értelmezve. Ez a nagymértékben limitált információfelhasználás ugyan összetettebb navigációt, útvonaltervezést nem tesz lehetővé, és komolyabb kognitív térkép kialakulását sem, de a kisebb számításigény a robotot reagensebbé teszi, és választ ad a szimuláció időosztásos tulajdonságából adódó kihívásra. A figyelem ilyen jellegű fókuszálása megfigyelhető különféle emberi ([71]) és állati ([41]) tevékenységeknél, és értelemszerűen intelligens ágensek tervezésénél is fontos szerepe lehet.

A képfeldolgozáson és a központi egységen kívül a többi modul mind a robot mozgását irányítja. Erre a célra többek között Braitenberg-jármű formájú modulok szolgálnak ([20]). Ezek az automaták szenzorokkal és motorokkal rendelkeznek, az egyszerű szabályozó mechanizmusok ezen elemek között foglalnak



(a) szeretetteli (b) felfedező (c) gyáva (d) agresszív

3.5. ábra. Elemi Braitenberg-járművek. A fent elhelyezkedő szenzorokon mért értékek a kapcsolatokon keresztül a megfelelő, alapesetben előre forgó kerekek mozgását fékezik vagy gyorsítják. Az érzetek asszimetriájából adódóan a járművek összetett mozgást végeznek.

helyet. Braitenberg különféle típusokat definiál, melyek egymástól a szenzorok és a motorok számában és kapcsolódásaikban térnek el. Az alaptípusoknál nincs meghatározva, hogy a szenzor a környezet mely tulajdonságát kíséri figyelemmel.

A járműveknek számtalan fajtája létezik, a talán legismertebb kétszenzoros és kétmotoros típusok abban különböznek egymástól, hogy melyik szenzor melyik motorral van összekötve, illetve, hogy a bemenet serkenti vagy gátolja-e a kimenetet. Ezek a lehetőségek négy típust definiálnak, melyek néhány emberi viselkedéshez való hasonlatosságuk miatt a következő neveket kapták: szeretetteli, felfedező, gyáva és agresszív (3.5. ábra). A szeretetteli és a felfedező járműveknél az inger a motoros választ az azonos, illetve az átellenes oldalon csökkenti. Ezért az első esetben a robot lassulva az ingerforrás felé fordul, és megáll előtte, míg a másodikban a forrástól elfordul, és az inger kis változásai is új keresésre bírják. A gyáva és az agresszív robotoknál az inger a motoros választ az azonos, illetve az átellenes oldalon növeli. A gyáva jármű emiatt ingerszegény irányba törekszik az ingerrel arányos sebességgel, míg az agresszív jármű gyorsulva közeledik a forrás felé.

A versenyzőnél használt modulok a fenti járművek speciális változatai. Az alapvető mozgás modulja esetében a robot nyolc infravörös szenzora (mint bemenet) és két motorja (mint kimenet) között alakul ki a fent definiált kapcsolat. A robot forrás és tárgy érzékelése hiányában egyenes vonalban halad, ezt a mozgást akadálykikerülés váltja föl fal megközelítésekor. A viselkedés leírását a 3.1. algoritmus adja, mely Michel kódja alapján készült ([42]). Az eljárás a felfedező és a

gyáva Braitenberg-járművek egyesítése, ahogyan az a BALSÚLY és a JOBBSÚLY skalártömb elemeiből látható. A növekvő negatív értékek a robot előtti tárgyaktól való elfordulást stimulálják. A kisebb pozitív számok az azonos oldali akadályoktól történő távolodást segítik. A hátsó szenzorok felhasználásával a robot nem tolat rá akadályokra.

3.1. algoritmus Az alapmozgás vázlata ([42] nyomán)

```
ALAPMOZGÁS()
   BALSÚLY[] := {4,4,6,-18,-15,-5,5,3}
   JOBBSÚLY[] := {-5,-15,-18,6,4,4,3,5}
   BALSEBESSÉG := SEBESSÉG.MAXIMUM
   JOBBSEBESSÉG := SEBESSÉG.MAXIMUM
   FOR(I := 0; I < 8; ++I) DO
      ÉRZET := TÁVOLSÁGSZENZOR(I)
      JOBBSEBESSÉG := JOBBSEBESSÉG + ÉRZET * JOBBSÚLY[I]
      BALSEBESSÉG := BALSEBESSÉG + ÉRZET * BALSÚLY[I]
   END FOR
END ALAPMOZGÁS
```

Amikor a robot mozgás közben nem észlel energiaforrást, akkor a körbefordulás — és ezen keresztül az energiaforrás keresésének — motivációja növekszik, ami a szimuláció ötven lépése után a robot 360 fokos körbefordulását okozza. Ha eközben forrás tűnik fel a képmezőben, az kijelöli az új haladási irányt.

3.2. algoritmus Az energiaforrás megközelítésének vázlata

```
KÖZELÍTÉS()
   BALSEBESSÉG := SEBESSÉG.MAXIMUM
   JOBBSEBESSÉG := SEBESSÉG.MAXIMUM
   FOR(X :=0; X < KAMERA_SZÉLESSÉG/2; ++X) DO
      IF KÉP[X] = FORRÁSSZÍN THEN
         BALSEBESSÉG := BALSEBESSÉG - (KAMERA_SZÉLESSÉG/2 - X)/10
      END IF
   END FOR
   FOR(X := KAMERA_SZÉLESSÉG/2; X < KAMERA_SZÉLESSÉG; ++X) DO
      IF KÉP[X] = FORRÁSSZÍN THEN
         JOBBSEBESSÉG := JOBBSEBESSÉG - (X - KAMERA_SZÉLESSÉG/2)/10
      END IF
   END FOR
END KÖZELÍTÉS
```

Energiaforrás látványa esetén egy megközelítő viselkedés veszi kezdetét, ami egy szeretetteli típusú Braitenberg-járműnek felel meg. A viselkedés annyiban speciális, hogy a kép széle felől jelentkező stimulust súlyozottabban veszi figyelembe, ahogy az a 3.2. algoritmusban is látható. Ez a fajta nem-linearitás segít abban, hogy a robot ne tévessze véletlenül szem elől az energiaforrást.

Az eddigi modulok segítségével elvileg minden energiaforrást meglelhet és megközelíthet a robot, mégis célszerű egy falkövető modult is bevezetni. Ezzel a robot egyrészt akadálykikerülést végez, másrészt könnyen elhaladhat a forrás aktív mezője előtt, amivel sikeresen feltöltődik, harmadrészt optimális esetben energiaforrásoktól mentes térrészből juthat át töltődésre alkalmas helyiségbe, egyfajta taktikai tervezést végrehajtva. A modul 25 lépésenként aktiválódhat, de csak akkor, ha fal közelében van a robot. A falkövetés a taktilis szenzorok használatával úgy állítja a hajtóerőket, hogy a robot mindig 2-3 centiméterre legyen a faltól, ez akadályok sarkánál a befordulást jelenti (3.3. algoritmus).

3.3. algoritmus A falkövetés vázlata

```
FALKÖVETÉS()
   BALÉRZET := (TÁVOLSÁGSZENZOR(0) + TÁVOLSÁGSZENZOR(1))/2
   JOBBÉRZET := (TÁVOLSÁGSZENZOR(4) + TÁVOLSÁGSZENZOR(5))/2
   IF BALÉRZET > JOBBÉRZET THEN
      BALSEBESSEG := BALÉRZET * 0.2 - 20
      JOBBSEBESSEG := JOBBÉRZET * -0.2 + 60
   ELSE
      JOBBSEBESSÉG := BALÉRZET * 0.2 - 20
      BALSEBESSÉG := JOBBÉRZET * -0.2 + 60
   END IF
FALKÖVETÉS
```

Vannak olyan esetek is, amikor a robot több falhoz is nagyon közel kerül, gyakorlatilag beszorul közéjük. Ekkor a menekülés kicsi modulja aktivizálódik, amely mindössze néhány lépésnyit tolat, hogy a kényes szituációból kimeneküljön a robot.

A robot mozgása összességében a következő elemekből áll. A falak elől kitérve egyenesen mozog, időről-időre körbefordul energiaforrást keresni. Ha észrevesz egyet, akkor megközelíti, és feltölti magát a robot, ha nem, akkor folytatja útját. Töltődés nélkül pedig egy idő után falkövetésbe kezd, és így jut el a terep új vagy régebben bejárt részeire.

3.4. Eredmények

A két verseny során a kiírásoknak megfelelően elkészítettem a fejezetben vázolt robotirányító algoritmus C és Java nyelvű változatát. A program mindkét versenyen jól teljesített a mintegy tucatnyi konkurens csapattal szemben. Első alkalommal a heti bemelegítő fordulók során, a fejlesztések hatására egyre javuló teljesítményt mutatott, majd a tényleges verseny körmérkőzéses szakaszában 100%-os lett, és csak a második helyezettel folytatott páros mérkőzésen maradt alul, azaz második lett. A második versenyt, melyen a kiírás szerint mindenki a rangsorban előtte állót hívta ki, a beküldött robot szinte végig az első helyen töltve nyerte meg. A győzelem az 1000 $-os fődíj és egy Webots szimulációs programcsomag elnyerését jelentette, mely a további robotszimulációs kísérletek eszköze lehetett.

Az eredményesség oka a ritka elakadás mellett a hatékony falkövetésben rejlett, aminek következményeként mérkőzésről mérkőzésre átlagosan 5 energiaforrást használt fel a robot, szemben a többiek 2-es átlagával. Ez az eredmény taktilis szenzorok intenzív, illetve fényszenzorok egyszerűsített használatával volt elérhető, ami nagyjából egy szinte teljesen vak ember tapogatózó keresésének felel meg.

A komolyabb ellenfelek közül az első verseny győztese a kamera teljes képét feldolgozta, majd rengeteg előzetes mérés alapján összeállított egy nagyméretű távolságtáblázatot, amiből térképet hozott létre. Bár a tárolt adatok miatt a forráskódja nagyjából nyolcszor akkora volt, mint a saját kontrolleremé, a hatékony térképolvasás és a jó útvonaltervezés elég volt a győzelemhez. A második verseny esetében a második helyezett az én robotomnál egyszerűbb felépítésű, falkövetés nélküli reaktív robot volt, az üres terekbe jutásra ezért kevesebb esélye lehetett. A a harmadik helyezett ezzel szemben egy foglaltsági hálón alapuló kognitív térképet hozott létre, de az útvonaltervezést már nem megfelelően valósította meg. A negyedik helyezett az első verseny győzteséhez hasonló képfeldolgozást végzett, ezúttal neurális hálót felhasználva.

3.5. Következtetések és további feladatok

A viszonylag egyszerű és nem igazán zajos környezet is kellően összetett feladatot jelentett, ami rengeteg meglepetést tartogatott a tervezőnek. Szinte minden környezetben adódtak olyan helyek — amelyek a körülmények együttállása miatt

— a robot elakadását okozhatták: keskeny átjárók, megfelelő sűrűséggel elhelyezett oszlopok, sőt az ötszög fölülnézetű energiaforrás hatékony megközelítése sem volt triviális.

A problémák másik csoportja túlmutat a szenzomotorikus szinten megoldható feladatokon, és egy közös forrása van: a robot kizárólag lokális döntéseket hoz. Globális döntéshez a környezet részletesebb ismeretére, valamiféle memóriára van szükség, ami optimális esetben a teljes környezetet felöleli, értelemszerűen abban az állapotában, ahogy a robot utoljára érzékelte. Ezt a célt szolgálná egy kognitív térkép létrehozása.

Az eredmények alapján látható, hogy megvalósítható térképkészítés nélkül is elfogadhatóan működő, sikeres tájékozódó eljárás, azonban a terep alapos megismeréséhez, általános célú feladatvégzéshez elkerülhetetlen egy teljes navigációs rendszer létrehozása. A versenyen nyert szimulátorprogram, azon kívül hogy lehetővé tette a további kutatást, motivációt is adott a navigáció témakörének alaposabb megismeréséhez.

A verseny eredményeként egy olyan hierarchia alakult ki, amelyben a robotok az előző szintre épülve egyre összetettebb viselkedésre képesek, és egyre sikeresebbek lehetnek a további versenyekben.

A viselkedési szintek a következők:

1. A robot képes akadályok elkerülésére és energiaforrások alkalomszerű megközelítésére.

2. A robot a teljes terep bejárására képes, véletlenszerű módon, de mindenhová eljuthat. Ezt a szintet érte el a fejezetben részletezett kontroller.

3. A robot térképet épít, amivel minden energiaforrást meg tud találni.

4. A robot az energiaforrások újratöltődését is figyelembe véve optimális útvonalat jár be, az energiafelhasználás maximalizálása érdekében.

5. A robot hatékonyan akadályozza ellenfelét feladata teljesítésében.

4

Navigáció foglaltsági hálóval

4.1. Bevezetés

Amikor gyermekkoromban megérkeztünk a család hétvégi telkére, kutyánk hosszú percekig csak azzal volt elfoglalva, hogy minden zegzugot bejárjon. Viszonylag kevés figyelmet fordított a megszokott növényekre és kerti bútorokra, de annál nagyobb érdeklődéssel szaglászta körül a friss vakondtúrásokat, és elégedetten ugatott, ha egy sündisznót is felzavarhatott.

Hasonló megfigyeléseket számos más élőlénynél leírtak. Székely és társai megállapították, hogy a paradicsomhalak egy sakktáblaszerűen felosztott akváriumban igyekeznek minél előbb az összes szobát bejárni, és azokban a részekben több időt töltenek, ahol valami érdekeset találnak ([178]). Csimpánzok esetében Menzel úgy találta, hogy a több ezer négyzetméteres terepen elhelyezett néhány új tárgyat is hamar megtalálják, a nem túl érdekeseket is alaposan megvizsgálják, és helyükre később is pontosan emlékeznek ([110]). Az élőhely alapos feltérképezése az állatok többségénél kritikus a túlélés szempontjából, így érthető, hogy komoly figyelmet fordítanak a környezet rendszeres felmérésére.

Az ember szintén bejárja és folyamatosan vizsgálja környezetét (annak legapróbb részleteit is) a hétköznapi feladatvégzések közben. Ilyen típusú munka a takarítás (porszívózás, padlótisztítás), fűnyírás, tárgyak összegyűjtése és egy terület őrzése, felügyelete. Ezen munkák automatizálásához az előre meghatározott környezet hatékony megismerésére van szükség.

Kutatásom célja a fenti állati és emberi viselkedések robotokra alkalmazása,

mely az alábbiak szerint foglalható össze. Az általam használt szimulált robotnak ismeretlen, a szenzorok érzékelési tartományánál lényegesen nagyobb környezetek teljes bejárását kell elvégeznie, azaz a terep minden egyes részére el kell jutnia. A világok falakkal határolt, vízszintes padlójú kísérleti terepek, ahol több-kevesebb geometrikus tárgy, illetve fal akadályozza a mozgást. A környezet megismerése során a robotnak el kell készítenie a terület kétdimenziós térképét is.

A feladat megoldására egy foglaltsági hálót alkalmazó térképezési eljárást készítettem Thrun munkája nyomán ([182]), ami a Webots mobilrobot-szimulátorban működik ([171], [175]). A térképet az ultrahangos szenzorok méréseiből megalkotott inverz szenzormodell szerint számítottam ki. A terepen végzendő navigációhoz a térképen alapuló saját értékiterációs eljárást hoztam létre. Ez egy költségmátrix segítségével az ismeretlen terek távolságát mutatja meg a robotnak, ami a távolságok alapján útvonalakat tervez és bejárja környezetét.

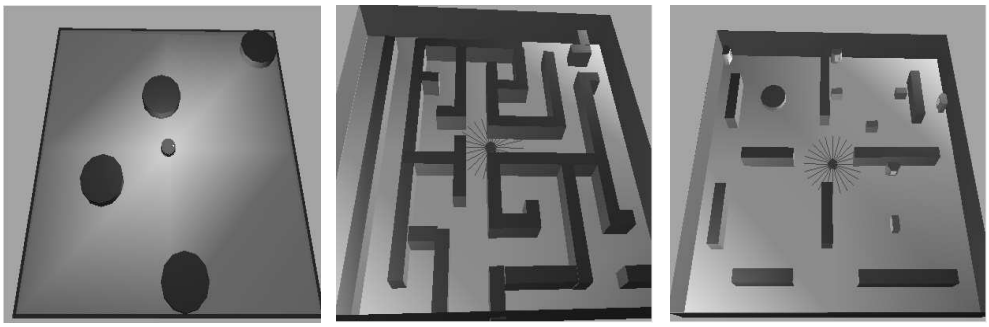
4.2. Foglaltsági háló készítése

A foglaltsági hálót létrehozó térképépítést és értékiterációs útvonaltervezést néhány négyzetméteres virtuális környezetekben végeztem el. A használt terepeket a 4.1. ábra mutatja be.

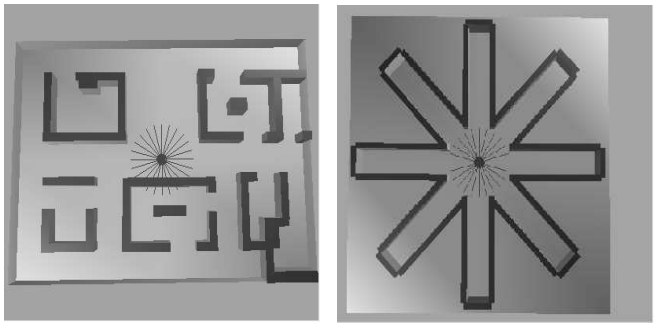
A választott tenyérnyi Khepera robot infravörös érzékelőit 24 darab, 15 cm hatótávú szonárra cseréltem, így a robot körül egyenletesen felosztott szenzormező alakul ki, mely kevésbé színérzékeny, mint az eredeti megoldás. Ezek a módosítások megkönnyítik a pontos térkép gyors létrehozását. A távolságszenzorok használata vizuális szenzorok helyett gyakran célszerű, mivel az érzetek dimenziószáma miatt a képfeldolgozás jóval összetettebb feladat, és valódi robotnál a kamerák felszerelése drágább is.

A foglaltsági hálón alapuló navigáció megvalósításának lényeges elemei a következők:

- szenzorinterpretálás
- időbeli egyesítés
- helymeghatározás
- a globális foglaltsági háló építése
- útvonaltervezés értékiterációval



(a) Nyílt terep (b) Labirintus (c) Irodaszerű környezet



(d) AAAI verseny (e) Sugaras pálya

4.1. ábra. A kísérleti környezetek középen a Khepera robottal. A robotból kiinduló, sugárirányú vonalak a szonárok érzékelési tartományát jelzik.

4.2.1. Szenzorinterpretálás

A szenzorok által érzékelt jelek értelmezése a foglaltsági háló készítésének első lépése.

A 24 darab, körkörösen elhelyezett ultrahangos szenzor kellően sűrű információforrás ahhoz, hogy a robot környezetében a foglaltságot számítani lehessen. Ehhez az *inverz szenzormodellt* kell megalkotni, ami esetünkben egy leképezés a mérések skalárértékeiről egy kétdimenziós valószínűségmátrixra, vagyis $p(occ_{x,y}|s)$ valószínűségeket kell meghatározni, ahol s egy mérés az (x, y) cellán. Az inverz szenzormodell elkészítésére többféle módszer is létezik: Moravec és Blackwell esetében a modell egy paraméteres függvény, melynek változóit egy hegymászóalgoritmus segítségével határozzák meg ([119]), míg Thrun backpropagációs neurális hálót használ az optimális leképezés kiszámítására ([182]). Én

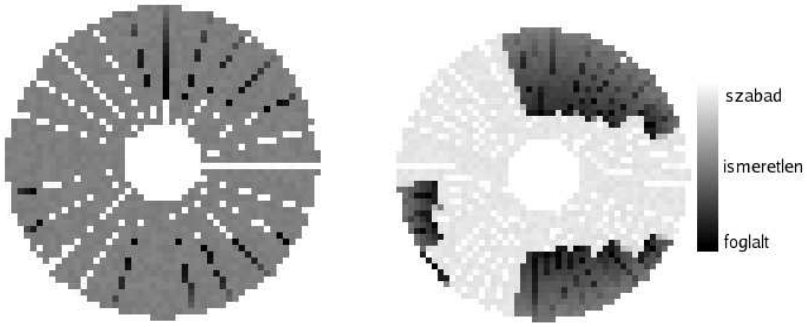
a szintén gyakori kézi hangolás mellett döntöttem, mivel a szimulátorban a szonár méréseit sokszor megzavaró tükröződés kevéssé jut szerephez, így a tesztek alapján a szonár kellően pontos lehet. A módszer hátránya, hogy eltérő visszaverődést mutató felületeket tartalmazó környezetben az eljárást újra kell hangolni.

Az inverz szenzormodell meghatározása a következőképpen történik. A szenzor által visszaadott érték az érzékelő specifikációja által meghatározott távolságot jelöli. Bár a szenzor zajos értékeket közvetít, a kapott távolságnál „közelebb" logikus alacsony valószínűséget (ε), a szenzorérték helyén magas valószínűséget ($1-\varepsilon$) rendelni; majd ezután, a bizonytalanságból adódóan, a valószínűség lecsökkenhet a teljes információhiányig ($\frac{1}{2}$). A 4.2. ábra bal oldala mutatja a mérések foglaltsági valószínűségre képezésének eredményét. A szenzorok sugárirányaiban a robot közelében világos árnyalatok láthatók, majd néhány irányban akadályt jelző sötétebb szakaszok következnek, végül ezek is bizonytalan szürkévé válnak. A sugarakon kívül a lokális térkép középszürkéje teljes bizonytalanságot jelez.

A vázolt modellt a továbbiakban kiterjesztettem. Mivel a cél a bejárható útvonalak megtalálása, ezért hasznos a szonár sugarát mesterségesen kiszélesíteni legalább a robot szélességére. Az approximációból adódó pontatlanságért kárpótol az információgazdagabb lokális — és később pontosabb globális — térkép létrehozásának lehetősége. A kiszélesítés során a sugarak mindkét oldalán két cellányi szélességben a mérések alapján számított távolságbecslést tároltam el. A lokális foglaltsági háló felbontásából adódóan ezáltal a robot közvetlen környezetének szinte minden cellájára vonatkozik direkt vagy indirekt távolságbecslés. A teljes szenzormodell által képezett lokális foglaltsági hálót a 4.2. ábra jobb oldala mutatja.

4.2.2. Időbeli egyesítés

A teljes foglaltsági háló a környezet bejárásával keletkezik. Ez annyit jelent, hogy sok mérést végez a robot egy adott cellára vonatkozóan is. Mivel a mérés körülményei — a robot helye és iránya, a zaj stb. — változnak, ezért különböző mérési értékeket kell egy helyre integrálni. Tehát az egyes mérésekből származó feltételes valószínűségekből — melyet $p\left(occ_{x,y}|s^{(t)}\right)$ jelöl a t időpontban — kell kiszámítani az összes mérésből adódó feltételes valószínűséget, vagyis $p\left(occ_{x,y}|s^{(1)}, s^{(2)}, \ldots, s^{(T)}\right)$-t. A számításhoz a Bayes-tételt ([115]) és a méré-



(a) Egyszerű lokális térkép (b) Bővített lokális térkép

4.2. ábra. A lokális térkép kiszélesítésének szerepe. Bal oldalon csupán a szenzorsugarak irányában láthatóak az ismeretlen terület szürkéjétől eltérő mérések. Jobb oldalon, a kiszélesített sugarak miatt, a mérések közötti területen közelítő foglaltsági valószínűségek jelennek meg.

sek egymástól való függetlenségének feltételét kell felhasználni, ami annyit jelent, hogy $p\left(s^{(t)}|occ_{x,y}\right)$ független $p\left(s^{(t')}|occ_{x,y}\right)$-től, ha $t \neq t'$.

Ezek alapján a következő számítási mód adódik az összes mérésen alapuló foglaltsági valószínűségre:

$$\Pi(x,y,T) = p\left(occ_{x,y}|s^{(1)},s^{(2)},\ldots,s^{(T)}\right) =$$

$$1-\left(1+\frac{p(occ_{x,y}|s^{(1)})}{1-p(occ_{x,y}|s^{(1)})}\prod_{\tau=2}^{T}\left(\frac{p(occ_{x,y}|s^{(\tau)})}{1-p(occ_{x,y}|s^{(\tau)})}\frac{1-p(occ_{x,y})}{p(occ_{x,y})}\right)\right)^{-1} \quad (4.1)$$

ahol $p(occ_{x,y})$ a kezdeti valószínűség, általában $\frac{1}{2}$ értéket vesz föl ([182]).

A 4.1. képlet jelentősége, hogy egyedi, egymástól független mérésekre alapozott valószínűségek kombinációjából áll elő oly módon, hogy az újabb és újabb értékek a korábbi számításokhoz illeszthetők, vagyis a mérések egyesítésével egy adott hely foglaltsága kiszámítható.

Ugyanakkor fontos megjegyezni, hogy a mérések függetlensége nem mindig teljesül, mégis gyakran fölhasználják, mert a térkép készítését nagymértékben egyszerűsíti ([183]). Leginkább akkor jelentkezik ez a probléma, ha a robot

álló helyzetben gyűjt össze rossz visszaverődésből keletkezett értékeket, vagyis hibás mérések sorozatát összegzi egy adott pozícióban. Esetünkben a robot folyamatosan mozog, és a visszaverődésből származó hiba sem jellemző a szonárra a szimulátorban, ezért a különböző időpontok méréseinek függetlensége feltehető.

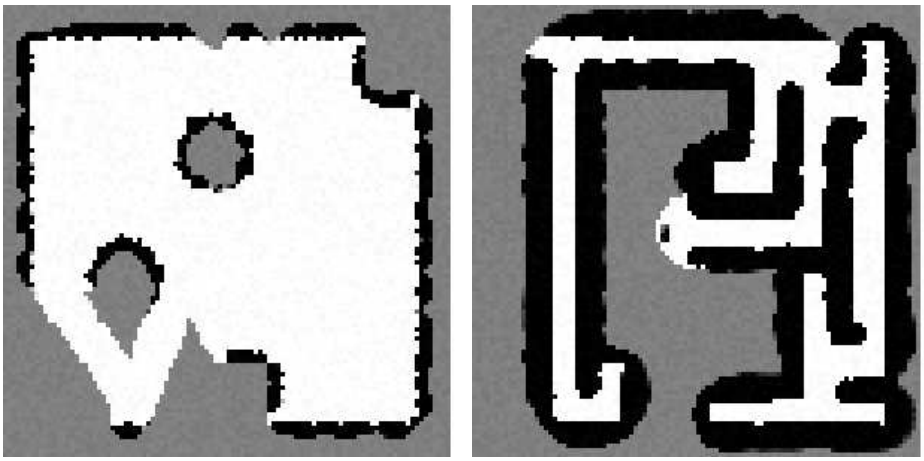
4.2.3. Helymeghatározás

A robot mindenkori pozíciójának meghatározásához, a szimultán lokalizáció és térképépítés problémájának megoldásához a szimulációs környezetben implementált felügyelő programot használtam. Ez a központi modul egy rádiós csatornán küldi el a robot számára az irány- és a helykoordinátákat, így választva ketté a lokalizáció és a térképezés feladatát.

Ennek a választásnak egyrészt az az oka, hogy alapvetően a térképkészítés és a térképhasználat folyamatára kívántam koncentrálni. Másrészről, pusztán kis hatótávolságú szonárra alapozva a robot pozícióját üres térrészekben, ahol huzamosabb ideig semmilyen tereptárgyat nem lehet érzékelni, a korábban bemutatott lokalizációs módszerek is nehézségekbe ütköznek. Egy lehetséges megoldás nagyobb hatótávú szenzorok, például egy kamera bevezetése, mellyel a terep sarkainál elhelyezett, mindig látható tereptárgyakra alapozott folyamatos triangulációt végezne a robot. Másik megközelítés lehet az odometria használata, ami különösen a szimulátorban működhet megbízhatóan, mivel a világmodellből a szisztematikus zajok kiküszöbölhetők, és csupán a véletlenszerű zajok okozhatnak problémát, melyek egy nagyságrenddel kisebb mértékben rontják az odometrikus becslést ([16]), mint a rendszeresek. Vagyis az odometria átsegíthetné a szonárra alapozott érzékelést a környezeti jellemzőktől mentes területen. Ugyanakkor ez a módszer valós robotba átültetve több utómunkát igényelne a szisztematikus zajok felmérése miatt.

4.2.4. Globális foglaltsági háló építése

Miközben a robot körbejárja a terepet, a lokális foglaltsági háló adatai a globális térképre kerülnek. Ez egyrészről polárkoordináták Descartes-koordinátákra transzformálását jelenti, másrészről a szenzorok által mért értékek egyesítése is ekkor történik. Az idő előrehaladtával a robot egyre nagyobb területet fedez fel, amint az nyílt terepen és a labirintusban (4.3. ábra) is látható.



(a) Nyílt terep foglaltsági hálója (b) Labirintus foglaltsági hálója

4.3. ábra. Két készülő foglaltsági háló. A sötét részek nagy, a világos területek kis foglaltsági valószínűséget jelentenek, a szürke területek feltérképezetlenek.

4.2.5. Útvonaltervezés értékiterációval

Bár a robot az eddigi modulokkal felkészült a térképkészítésre, szükség van valamilyen hajtóerőre, aminek hatására bejárja a teljes terepet, egyébként csupán véletlenszerűen mozogna.

Ennek érdekében egy dinamikus programozási eljárást alkalmaztam. Ez az eljárás a problémát — vagyis, hogy a térkép mely része felé érdemes elindulni — részproblémákra osztja, és a részek megoldásából állítja elő az optimális megoldást ([38]). A használt eljárás a Sutton és Barto könyvében is ismertetett értékiteráció Thrun által foglaltsági hálóra létrehozott változatának saját módosítása ([166], [182]). Ez az algoritmus a háló cellái alapján meghatározza annak a költségét, hogy az adott cellától milyen messze van a legközelebbi feltérképezetlen terület, és ezt egy költségmátrixban tárolja. A számításhoz a mátrix szomszédos celláinak értékét használja föl, vagyis rekurzív módon bontja részproblémákra a feladatot, és a kalkulációt elég kicsi hibáig ismételten elvégzi. A módszer a navigációs módszereket ismertető fejezetben bemutatott fix dekompozíciós útvonaltervezéshez sorolható.

1. Az eljárás első lépése a költségmátrix inicializálása. A nem bejárt cellák értéke 0, míg a bejártaké ∞ lesz:

$$V_{x,y} \leftarrow \begin{cases} 0 & \text{ha } (x,y) \text{ bejáratlan} \\ \infty & \text{ha } (x,y) \text{ bejárt} \end{cases}$$

2. A főciklus során minden bejárt cella értéke újraszámolódik. A valószínűleg foglalt cellák 1 költségértéket kapnak. A többi cella értéke a környező cellák költségének és foglaltsági valószínűségének minimumából adódik. A δ tag az út hosszúságát bünteti.

   $\forall$ bejárt (x,y)-re:

$$V_{x,y} \leftarrow \begin{cases} 1 & \text{ha } p\left(occ_{x+i,y+j}\right) > 1-\varepsilon \\ min(1, \min\limits_{\substack{i=-1,0,1 \\ j=-1,0,1}} \left\{V_{x+i,y+j} + p\left(occ_{x+i,y+j}\right) + \delta\right\}) & \text{egyébként} \end{cases}$$

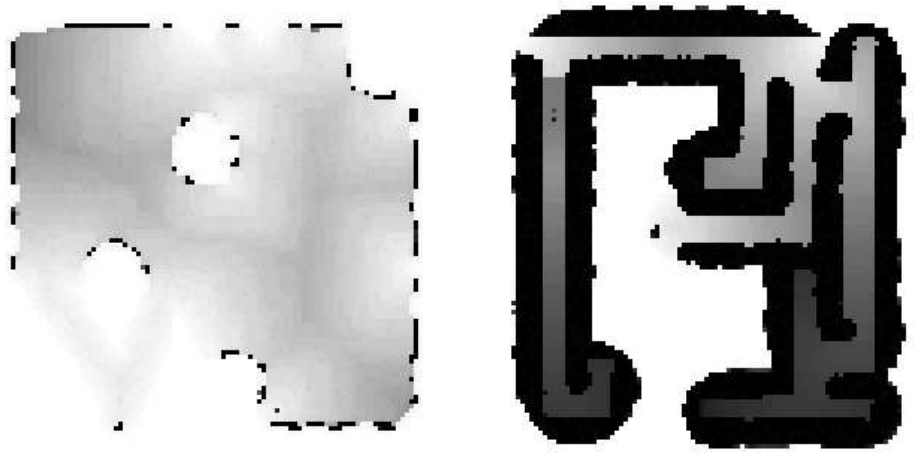
   Konvergencia után a V mátrix a kumulált költségeket tartalmazza. A számításigény csökkentése érdekében úgy találtam, hogy érdemes a konvergencia előtt befejezni az iterációt, amikor már elég kicsi a hiba, vagy a mátrix értékei kellően kis mértékben változnak. Az én megközelítésemben a költségmátrix méretének megfelelő számú iterációt végzek, mivel a tapasztalatok alapján már ez is elég pontos költségértékeket jelent.

   A konvergencia segítése érdekében alkalmazom az út hosszúságának büntetését, illetve magas foglaltsági valószínűség esetén a rekurziót leállító 1 értéket.

3. Mivel egy terület felfedezése az egész költségmátrixra kihat, ezért minden lépés után a teljes mátrixot újra kellene számolni. Ez azonban jelentősen lassítaná a program működését. Ezért alapvetően a robot egy 10 centiméteres környezetében működik az iteratív eljárás, ahol az ultrahangos szenzorok rövid távon érdemben változtatnak a foglaltsági háló értékein. A teljes mátrix a szimuláció minden 350. lépésében frissül.

A 4.4. ábra a kialakult költségmátrixokat mutatja a bejárás egy adott pontján a 4.3. ábrán bemutatott foglaltsági hálókhoz.

A robot navigációja ezután akadálykikerülésből és a költségmátrixra alapozott útvonaltervezésből tevődik össze. A környezet akadályait, a falakat és egyéb



(a) Nyílt terep költségmátrixa (b) Labirintus költségmátrixa

4.4. ábra. Az értékiteráció során előálló költségmátrixok a foglaltsági háló alapján számítva. A még nem bejárt területek „lámpaként” világítanak és vonzzák a robotot.

objektumokat a Braitenberg-járműveknél megismert felfedező és gyáva típusok kombinációjára alapozott mozgás segít kikerülni, a tárgyak közelségével hatványozottan növekvő taszítóerő felhasználásával. Az új irány meghatározásához a robot igyekszik a folyamatosan frissített kis környezetben található minimális érték felé mozogni, hisz abban az irányban érdemes nem bejárt cellák után nézni. A kis környezet segítségével az olyan cellák felé történő próbálkozás kikerülhető, amivel a robot nincs közvetlen összeköttetésben. A robot aktuális orientációja szintén szerepet játszik a felfedezés irányának meghatározásában: a hasonlóan kis költségű irányok közül azt választja a robot, amelyik leginkább az aktuális haladási irányba esik. Ezzel biztosítható a folytonos, nem csapongó mozgás. Az egymást kiegészítő minimumkeresés és akadálykikerülés együttes használata a robot elakadás nélküli navigációját teszi lehetővé.

4.3. Eredmények

A fejezetben bemutattam egy általam készített foglaltsági hálón alapuló értékiterációs robotirányító eljárást. Ismert eljárásokat alapul véve létrehoztam a szimulált környezet támasztotta igényeknek megfelelő inverz szenzormodellt és értékiterá-

ciós módszert. A navigációs algoritmus öt különböző környezetben, három különböző kiindulási pontból, a robot mozgását meghatározó öt eltérő pszeudovéletlenszám-generátor kezdőértékkel indított futtatás során bizonyította képességeit. A környezeteket oly módon választottam ki, hogy azok lehetőleg a térképkészítés és a bejárás közben felbukkanó problémák minél szélesebb spektrumát fedjék le.

A környezetek az alábbiak voltak: egy nyílt terep néhány kör alaprajzú tárggyal (4.1. ábra (a) rész), egy labirintus ((b) rész), egy irodaszerű környezet, mely az Artificial Life Creators Contesten is szerepelt ((c) rész), egy terep, melyet az 1994-es AAAI autonóm mobil robotok versenyén használtak ((d) rész, [182]) és egy sugaras labirintus, melyhez hasonlót kognitív térképpel kapcsolatos kutatásokban gyakran alkalmaznak ((e) rész, [133]). A nyílt terep 1 m^2, az AAAI labirintus 1.85 m^2, míg a többi környezet 2.25 m^2 nagyságú.

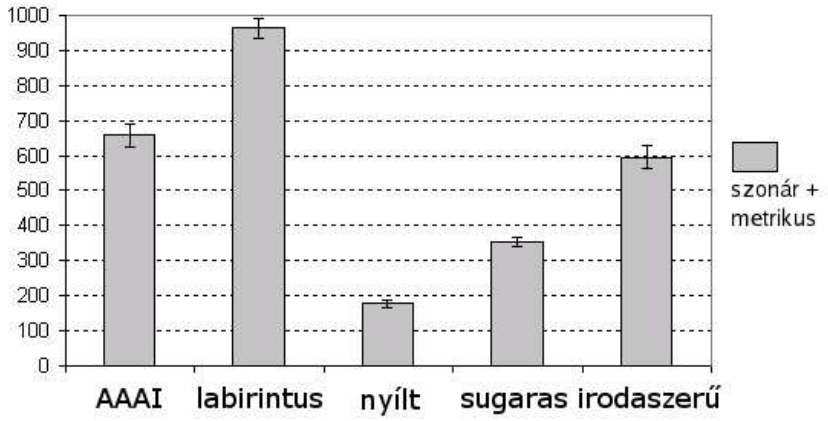
Az ultrahangos szenzor méréseit mintegy 20%-os, szimmetrikus, egyenletes eloszlású zaj terhelte. Az akadálykikerülés paramétereinek alapos finomhangolása után a robot minden terepet sikerrel bejárt, és a foglaltsági térképeket elkészítette.

A kísérleti futások időeredményének pályánkénti átlagát a 4.5. ábra tartalmazza. Teljesítménymértékként a 90%-os felderítéshez szükséges időt használtam. A teljes felderítés helyett ez az érték jobbnak tűnt, mivel így a futás végén felerősödő véletlen hatás (így például a félig takarásban lévő sarkokhoz való visszatérés) kiküszöbölhető. A másodpercben mért értékek a valódi Khepera teljesítményének felelnek meg. A valós robottal történő összehasonlítást a Webots szimulátor 4-es verziója teszi lehetővé, mely a futási sebességet képes szabályozni a valóságnak megfelelő értékre[1].

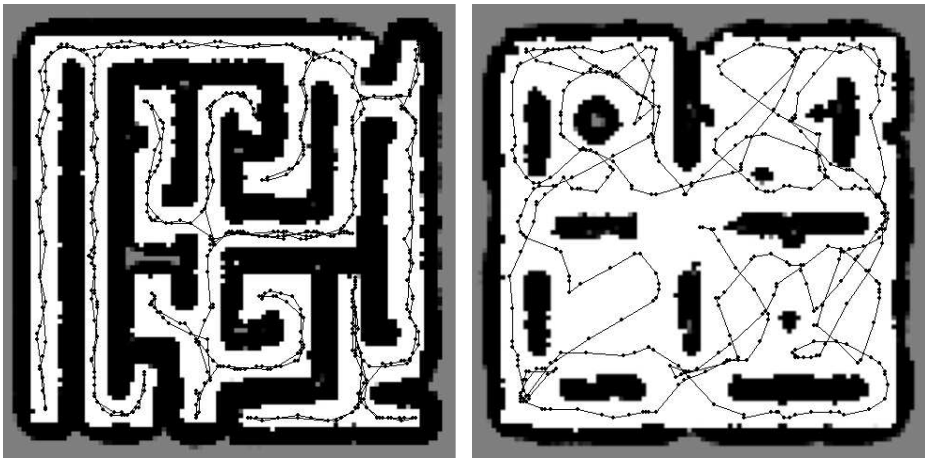
Az időeredmények alapján látható, hogy a kisebb, egyszerűbb terepekkel hamarabb végez a robot (nyílt és sugaras környezet). Az egyforma méretű pályák közül pedig a nagy üres térrészekkel rendelkezőt preferálja (irodaszerű), a hosszú folyosókkal és sok szobával szemben (AAAI verseny és labirintus).

Az eljárás működésébe enged bepillantást a 4.6. ábra, mely a robot trajektóriáját mutatja be egy-egy kiválasztott futás esetében a labirintusban és az irodaszerű környezetben. A két terep közötti különbség első ránézésre szembetűnő: a labirintus sokkal kötöttebb útvonalat engedélyez a robotnak, mint az irodaszerű környezet, ahol az üres térrészeken szabad a mozgás. Az előbbi esetben jól ki-

[1]Ez természetesen csak a valódi robotnál gyorsabb számítógép esetén használható. Enélkül a művelet nélkül a szimulátor mindig a futtató számítógép által aktuálisan elérhető legnagyobb számítási kapacitáson dolgozik, ami a futási idő összegzését nehézkessé teszi.



4.5. ábra. A terepek bejárásához szükséges idő másodpercben, ahogy az a valódi Khepera robotban mérthető lenne. Minden egyes oszlop egy környezetet jelöl, és 15 futás átlagos idejét tartalmazza három különböző kiindulási pontból, 95%-os konfidenciaintervallummal.



(a) Labirintus (b) Irodaszerű környezet

4.6. ábra. Az értékiterációs navigációt alkalmazó Khepera trajektóriája két terepen, az elkészült foglaltsági hálóra illesztve. A vonalon megjelenő pontok a robot helyét jelölik másodpercenként.

rajzolódik, ahogy a robot elmegy a folyosók végéig, majd nagyjából ugyanazon az úton visszamegy. Észrevehető, hogy a hosszú, egyenes folyosókon sem halad teljesen célirányosan a robot, az útvonaltervezés az oldalsó falakat érintve irá-

nyít. Az irodaszerű környezetet a falak terelő hatása nélkül is bejárja a robot, nem hagyja egyik részt sem feltérképezetlenül. Az is megfigyelhető, hogy a négy fő térrészen a robot nem egymás után megy végig, hanem egy találkozási ponthoz érve könnyen az új területen folytatja útját, és később visszatér a befejezetlen részekhez, ami nem feltétlenül optimális.

4.4. Kapcsolódó munkák

A foglaltsági hálót először Moravec és Elfes fejlesztették ki, mint egy olyan térképezési eszközt, mely az érzékelésben rejlő bizonytalanságot robusztus módon, valószínűségekkel kezeli, s mely inkrementális módon frissíthető és navigációra is alkalmazható ([120], [51]). Az eljárást később kamerákat használó háromdimenziós környezeti térkép építésére is kibővítették, melynél a kezelendő objektumok száma ezresről a milliós nagyságrendbe lépett ([106]). A tervek szerint 2010-re olyan robotok tömeges gyártására kerülhet sor, melyek navigációs eljárását a három dimenziós foglaltsági háló adja[2]. Egyszerűbb, porszívózásra alkalmas típus már most is kapható[3], bár bevallottan még csak oktatási és hobbi célokra, mivel az irányítást egy számítógépes összeköttetésen keresztül a felhasználónak kell végeznie ([11]).

A foglaltsági háló — jórészt egyszerű kezelhetősége miatt — igen népszerűvé vált, ezért számtalan robot esetében használják térképezési eljárásként, különféle szenzorokat alkalmazva, így szonárt és kamerát együttesen ([184]), sztereó kamerát ([121]), szonárt és lézeres távolságmérőt ([48]) vagy infravörös érzékelőt.

A foglaltsági háló önmagában egy környezetreprezentációs módszer és nem egy teljes navigációs eljárás. Az útvonaltervezés elvégzése közvetlenül a foglaltsági hálón már a kezdeti munkákban is megjelenik, így Elfesnél is, aki szemben az én értékiterációs módszeremmel, A^* algoritmus segítségével juttatja el robotját egyik pontból a másikba ([51]). Thrun értékiterációs módszere az ismeretlen részek felkutatására való ösztönzésként ([182]) nem tér el lényegesen az általam megvalósítottól, viszont ő a szimulátort csupán az inverz szenzormodell felépítéséhez adatgyűjtésként használta, egyébként valódi robottal dolgozott. Murray és Jennings a foglaltsági hálót szonárok helyett sztereó kamerák segítségével állították elő. A valódi robotban használt értékiterációs eljárást potenciamezőkre

[2] H. Moravec honlapja: http://www.frc.ri.cmu.edu/users/hpm/

[3] http://www.personalrobots.com/

alapozott útvonaltervezéssel kombinálták, hogy a taszítóerő segítségével az akadályokat kikerüljék ([121]). Szimulációs környezetben készített foglaltsági hálót Stepan és társai alkalmaztak, valódi robot mellett a Webots-ban is, módszeremmel szemben azonban ők monó kamera képe alapján készítették el a hálót, és az útvonalat egy potenciamezőt használó eljárással alakították ki ([163]).

4.5. Következtetések

A kutatás során bebizonyosodott, hogy a foglaltsági háló nem véletlenül kedvelt környezetreprezentációs módszer. Viszonylag könnyen létrehozható és iteratív módon karbantartható. A zajra általában kevéssé érzékeny, az újabb mérések segítségével a térkép többnyire egyre pontosabbá válik. Az eredmény az ember számára is könnyen értelmezhető, így a robot működése átláthatóbb. További előnye, mely későbbi feladatoknál hasznos lehet, hogy változó környezetek kezelésére részlegesen alkalmazható, illetve háromdimenziós modellként is megállja a helyét.

A foglaltsági háló általam is tapasztalt hátránya, hogy igen memóriaigényes, az objektumalapú térképekhez képest akár nagyságrendekkel is eltérhet. A háló a terepről kevéssé flexibilis térképet készít, az üres terek felbontása megegyezik a tárgyakkal zsúfolt, ezért fontosabb részek felbontásával. Az eljárás önmagában nem oldja meg a pozícióbecslést, így mindig kiegészítő lokalizációs algoritmusra van szükség. Egy további hátulütő a mérésekben tapasztalható zaj egymástól vett függetlenségének feltételezése, mely nem minden esetben adott.

Az értékiteráció előnyös tulajdonsága, hogy a számolás tetszőleges ponton megállítható, és már a részleges eredmények alapján is tervezhető útvonal[4]. A költségmátrix megfelelően módosított inicializálásával nemcsak a bejáratlan területet lehet megkeresni, hanem a teljes felfedezés után is megadható új cél. Mivel az eljárás minden ismert pontra kiszámítja a legközelebbi ismeretlen részhez vezető út költségét, ezért a robot egy eltévedés után is folytathatja tevékenységét. A teljes számítás további előnye, hogy ha több robot együttműködve végez felderítést, akkor a célok a költségmátrix ismeretében feloszthatók.

Az értékiteráció hátránya a nagy memória- és számításigény, ami a környezetek növekedésével arányosan egyre inkább érvényesül. Ezen kívül, miután a robot

[4] Vagyis az értékiteráció anytime algoritmus.

egy kis környezetben keresi a költségmátrix minimális értékét, ezért globális tervezésre ez a megoldás nem igazán alkalmas, a robot útvonala csupán lokálisan lesz optimális.

A kísérletek egy további tapasztalata, hogy sok szempontból kifizetődő egy szimulátor használata. A valós, fizikai kísérletekhez képest szimulátorral lényegesen egyszerűbb új kísérleti környezeteket létrehozni, módosítani a robot felépítését, ideértve szenzorainak számát, elhelyezkedését, modalitását és érzékenységét.

5

Navigáció topológiai gráffal

5.1. Bevezetés

A hatékony navigáció elképzelhetetlen valamilyen környezeti térkép nélkül. Továbbá a térképhez választott reprezentáció és navigációs módszer kulcsfontosságú, mivel nagyban befolyásolja, hogy miként helyezi el magát az ágens a világban, és hogy milyen terveket tud készíteni.

Ezt az élővilág példái is igazolják. Dolgozó hangyákat a fészek és az ételforrás közötti útvonalról áthelyezve egy ismeretlen területre, az állatok irányváltoztatás nélkül folytatják útjukat, ami azt sugallja, hogy egyszerűen vakon navigálnak ([32]). Ezzel szemben a tengeri gébek a dagály elvonultával gyakran a part menti mélyedések kis tavacskáiban maradnak, majd alkalomadtán pocsolyáról pocsolyára ugrálva jutnak vissza a tengerbe. Ehhez korábbi bejárások során elkészített háromdimenziós térképet használnak ([4]). Az összetettebb modell a környezet erőforrásainak hatékonyabb kihasználását teszi lehetővé.

A korábbi kísérletek során beigazolódott, hogy bár a foglaltsági hálón alapuló értékiterációt végző navigáció működőképes, nem feltétlen az optimális adaptációs megoldás, mivel amellett hogy számításigényes, azzal az előfeltevéssel él, hogy a környezet rácsszerű struktúraként leírt geometriai jellemzői a meghatározóak. Ennél a megoldásnál jobban igazodik az emberi gondolkodáshoz a topológiai navigáció, mely a kitüntetett helyeket és a közöttük lévő kapcsolatokat hangsúlyozza, a mozgás lehetőségét az egyik pontból a másikba, az akadályok és a veszélyek elkerülésével egyidejűleg. Az állatok számára is fontosabb a rej-

tekhely, az ivóvíz, az élelem, a fajtársak és a ragadozók közötti útvonalak hozzávetőleges ismerete azok centiméterre pontos, de egymással további relációban nem lévő meghatározásánál. Ezzel összhangban állnak azok a kutatások, melyek szerint az emberben lévő kognitív térkép egy többrétegű hierarchikus szerkezet, melynek legfelső szintje helyek közötti relációkon alapul ([154]).

Ezen megfontolások alapján célom az volt, hogy a Webots szimulátorban a diszkrét térreprezentációt jelentő foglaltsági hálóra építve — az értékiterációs útvonaltervezés helyett — létrehozzak egy topológiai alapú kogníciós réteget, melynek használatával a navigáció hatékonyabbá válik. A két módszer egyesítésével azok előnyös tulajdonságai találkozhatnak: a nagy felbontású térképről nem maradnak le fontos részletek, ugyanakkor az útvonaltervezés csak a robot leendő trajektóriáját komolyan befolyásoló elemek terében dolgozik.

A feladat megvalósítása érdekében a foglaltsági háló alapján vékonyítási eljárással elkészítettem az üres területek struktúráját megőrző vázat. Ezt a szkeletont láncolás és optimalizálás után alakítottam át a bejárható területek gráfjává. A robot kognitív térképe így vált hierarchikussá, melyből a topológiai gráf közvetítő közeget képez a diszkrét reprezentáció és a robotirányító eljárás között ([167], [173]).

5.2. Topológiai gráf készítése foglaltsági hálóból

A környezet topológiai gráfja az előző fejezetben ismertetett foglaltsági háló kiterjesztéseként jön létre. A foglaltsági háló tekinthető úgy, mint a környezet egy kétdimenziós szürkeárnyalatos, raszteres képe. Erre a képre a digitális képfeldolgozás eljárásai alkalmazhatók ([51]). A topológiai gráf létrehozásához szükséges *vektorizáció* többféleképpen végezhető el: a szkeletonizáció mellett elfogadott a kontúrok illesztése[1] vagy a ritka pixelvektorizáció[2] is ([187]). Az illesztéses módszer hátránya, hogy az egyenes vonalakat tartalmazó térképeket preferálja a komplex alakzatokkal szemben. A ritka pixelvektorizációnál a keresendő minták méretének megválasztása okozhat nehézséget. A három eljárás közül én a a szkeletonizációt választottam, melynél a zajérzékenység jelenthet problémát, ugyanakkor viszonylag egyszerűen számítható.

[1] contour matching
[2] sparse pixel vectorization

Tombre és társai szerint elegendő digitális képfeldolgozási algoritmus létezik, ezért nem érdemes újakat megalkotni, ennek megfelelően én is ismert eljárásokat használok ([188]). A vektorizáció során figyelembe vett másik fontos szempont az, hogy nem a legjobb, hanem az elfogadható minőségű vektorhalmaz létrehozása a cél, a lehető legrövidebb idő alatt.

A fenti választásból következően a foglaltsági háló topológiai gráffá történő átalakítását az alábbi lépések során végeztem el:

- szkeletonizáció
- a váz láncolása élekké
- a gráf optimalizálása
- útvonaltervezés topológiai gráffal

5.2.1. Szkeletonizáció

Az első feladat a bejárt és nem foglalt terület vázának létrehozása. Az eredményül kapott alakzat pontjai a környezet azon helyeinek felelnek meg, ahol a robot biztonságosan közlekedhet, mert középpontját a pixelek által kijelölt területen tartva valószínűleg egyetlen objektumban sem akad el.

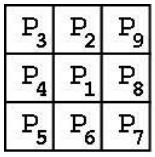
A váz készítéséhez első megoldásként a központi tengely transzformációt[3] használtam föl ([19]). A kiinduló alakzat egy belső pontja, akkor tartozik a központi tengelyhez, ha a körvonal kettő vagy több pontjától egyenlő távolságra esik. Sajnos a központi tengely transzformáció egy hátulütője a tesztek során kiderült: nem folytonos alakzatok esetén — amilyen a leképezendő diszkrét foglaltsági háló — az eredmény szaggatott lehet. Ez a hiányosság azonban nem elfogadható, mivel a váznak egy összefüggő gráffá kell válnia, hogy a bejárás teljes lehessen.

Másodjára a központi tengely transzformáció helyett egy vékonyító algoritmust alkalmaztam, mely iteratív módon egy pixel vékony vázzá zsugorította a kiinduló alakzatot ([78]). Az eljárás során a kezdeti pixelhalmazt „hagymaként meg kell hámozni”, azaz a határoló pixelek törölhetők oly módon, hogy eközben az objektum topológiája és morfológiája nem változik, vagyis egyetlen pixel sem tűnik el az egyenesek végpontjánál és a régiók találkozási pontjánál.

A vékonyítás az 5.1. algoritmusban leírtaknak megfelelően működik. A képpontok címkézését a P_1 pixel körül az 5.1. ábra mutatja. $Z0(P_1)$ jelöli a nulláról

[3] medial axis transform

nem nulla értékre váltások számát a $\{P_2,P_3,P_4,P_5,P_6,P_7,P_8,P_9,P_2\}$ körüljárás során. $NZ(P_1)$ pedig P_1 nem nulla értékű szomszédjainak számát adja meg. Az eljárás az összetett feltétel felhasználásával azokat a pixeleket törli, melyek a csökkentendő alakzat határoló felületén helyezkednek el, sok szomszédjuk van, és ezek egyúttal a képpont azonos oldalán vannak, vagyis a törlés hatására a váz nem szakad ketté.

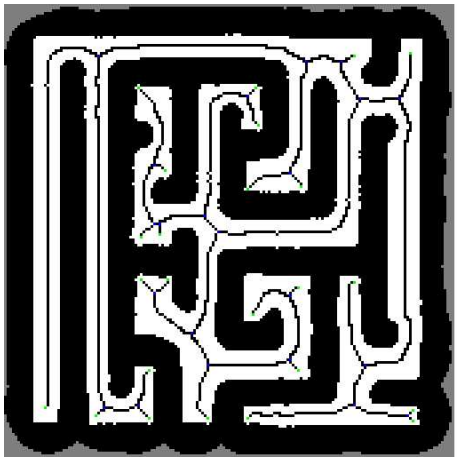


5.1. ábra. Pixelek címkézése vékonyításkor

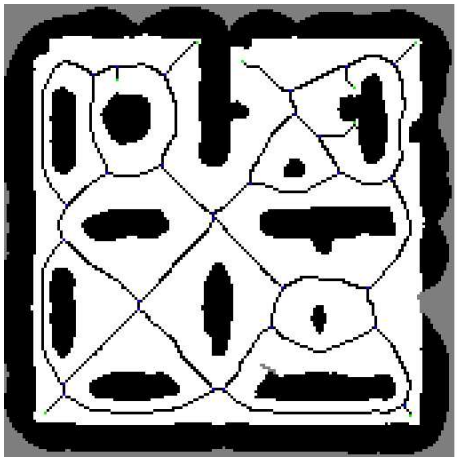
5.1. algoritmus A vékonyító algoritmus vázlata ([40] nyomán)

```
VÉKONYÍTÁS(KÉP,VÉKONYÍTOTT_KÉP)
   VÉKONYÍTOTT_KÉP := KÉP
   FOLYTAT := IGAZ
   WHILE FOLYTAT DO
      K := VÉKONYÍTOTT_KÉP
      FOLYTAT := HAMIS
      P1 := ELSŐ_PIXEL(K)
      WHILE P1 <> NULL DO
         IF 3 ≤ NZ(P1) ≤ 6 AND
              Z0(P1) = 1 AND
              (P2 * P4 * P8 = 0 OR Z0(P2) ≠ 1) AND
              (P2 * P4 * P6 = 0 OR Z0(P4) ≠ 1) THEN
            TÖRÖL(VÉKONYÍTOTT_KÉP,P1)
            FOLYTAT := IGAZ
         END IF
         P1 := KÖVETKEZŐ_PIXEL(K)
      END WHILE
   END WHILE
END VÉKONYÍTÁS
```

Az 5.2. és az 5.3. ábra a vékonyító algoritmus által előállított váz egy-egy példáját mutatja a foglaltsági hálóra helyezve. Az eredmény egy képpont vékony, egymásba csatlakozó pixelsávok struktúrája.



5.2. ábra. A labirintus váza



5.3. ábra. Az irodaszerű környezet váza

5.2.2. A váz láncolása

A bejárt és nem foglalt terület vázán való navigáció lehetséges, és valószínűleg hatékonyabb, mint az értékiteráción alapuló számítás, mivel egy strukturáltabb alakzaton kell útvonaltervezést végezni. Mégis érdemes a szkeletont további feldolgozás alapjául használni.

A váz ebben a formájában egy egységnyi vékony pixelhalmaz, amit az 5.2. algoritmusnak megfelelően gráffá transzformáltam. Az eljárásról bővebben Tombre és társai írnak ([187]). Az átalakításhoz először is a pixelsávok csatlakozási pontjait kell megtalálni. Ahol három pixellánc összefut az egy csúcspont, vagyis folyosók, térrészek találkozási pontja. Ennek megállapításához az algoritmus a vékonyításnál szereplő $Z0$ függvényt használja, mely a vizsgált pixel körüli nulla-nem nulla váltások számát adja vissza, és kereszteződést jelez, ha ez az érték eléri a hármat.

A csúcsok meghatározása után az algoritmus az összes csúcsból induló összes pixelsorozaton végighalad a szomszédos csúcsok, illetve a váz végpontjainak meghatározásához. Ennek során a lánc következő elemét négy, majd nyolc szomszédság, azon belül pedig csúcs, majd nem csúcs sorrendben keresi az aktuálisan vizsgált pixel szomszédjai között.

Mivel az algoritmus azokat a képpontokat definiálja csúcsként — szemben az én megközelítésemmel —, melyeknek legalább kettő szomszédja van, ezért az eljárást két ponton is módosítani kell, mert speciális esetekben, hibás gráf jönne létre. Az 5.4. ábrán a lánc létrehozása az „o”-val jelölt csúcsokból indul, és be-

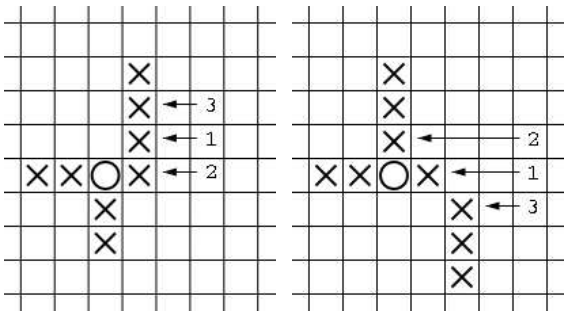
5.2. algoritmus A láncoló algoritmus vázlata ([187] nyomán)

```
LÁNCOLÁS(VÉKONYÍTOTT_KÉP,LÁNCHALMAZ)
   CSÚCSOK := CSÚCSMEGHATÁROZÁS(VÉKONYÍTOTT_KÉP)
   CSÚCS := ELSŐ_CSÚCS(CSÚCSOK)
   WHILE CSÚCS <> NULL DO
      SZ := ELSŐ_SZOMSZÉD(CSÚCS)
      WHILE SZ <> NULL DO
         LÁNC := ÚJLÁNC(CSÚCS,SZ)
         IF CSÚCS(SZ) THEN
            FOLYTAT := HAMIS
         ELSE
            FOLYTAT := IGAZ
            TÖRÖL(SZ)
         END IF
         WHILE FOLYTAT DO
            IF (SZ2 := KERES_NEM_CSÚCS_4SZOMSZÉD(SZ)) = NULL THEN
               IF (SZ2 := KERES_CSÚCS_4SZOMSZÉD(SZ)) = NULL THEN
                  IF (SZ2 := KERES_NEM_CSÚCS_8SZOMSZÉD(SZ)) = NULL THEN
                     SZ2 := KERES_CSÚCS_8SZOMSZÉD(SZ)
                     FOLYTAT := HAMIS
                  END IF
               ELSE
                  FOLYTAT := HAMIS
               END IF
            END IF
            ILLESZT(SZ2,LÁNC)
            SZ := SZ2
            TÖRÖL(SZ2)
         END WHILE
         HOZZÁAD(LÁNC,LÁNCHALMAZ)
         SZ := KÖVETKEZŐ_SZOMSZÉD(CSÚCS)
      END WHILE
      TÖRÖL(CSÚCS)
      CSÚCS := KÖVETKEZŐ_CSÚCS(CSÚCSOK)
   END WHILE
END LÁNCOLÁS
```

járja a környező „x”-szel jelölt pixeleket. A nem csúcsként azonosított pontok az algoritmus szerint törlendőek, miután egy lánc részévé váltak.

Az első probléma olyan esetekben merül föl, ami az ábra bal oldalán megjelenítetthez hasonló. Miután az eljárás a csúcs egy tetszőleges szomszédját választja,

ezért akár az 1-essel jelölt pixelnél is kezdhet, amit egyúttal töröl is. Ezután a 2-es és a 3-as képpontok elszakadnak egymástól, a lánc pontatlanul készülhet csak el.



(a) 1. probléma (b) 2. probléma

5.4. ábra. Láncolási problémák Tombre és társai algoritmusát követve. A megjelenített szituációkban az eredeti eljárás hibás gráfot is létrehozhat, ha a számozásnak megfelelő sorrendben választ következő képpontot.

A másik probléma az ábra jobb oldalához hasonló esetben jelentkezik. Ha a lánc létrehozása az 1-es képponttal kezdődik, akkor a keresésnek a következő lépésben a 3-as pixel felé kell fordulnia. Azonban az algoritmus nem tartalmaz semmi erre vonatkozó megszorítást, így a láncolás akár a 2-es pixel felé is folytatódhat, ami a 3-es képpontot kapcsolódás nélkül hagyja.

Az algoritmust a problémák ismeretében módosítottam: a kezdeti választáskor a négy szomszédság szerinti pixeleket preferálom, illetve csúcs melletti pixelek esetén a négy szomszédság szerinti szomszédokat ideiglenesen törlöm. A bővítések után a láncoló eljárás már elő tudta állítani a navigációs gráf kezdeti változatát, melyben a váz kereszteződési pontjai és a végpontok a gráf csúcsai, a köztük lévő összeköttetések pedig a gráf élei.

5.2.3. A gráf optimalizálása

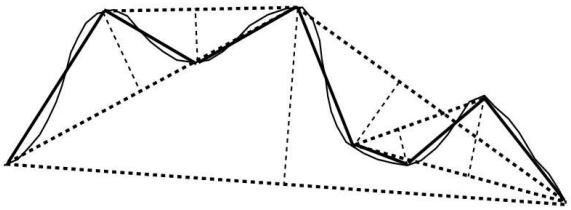
A gráf első változata nem alkalmazható közvetlenül útvonaltervezésre, mivel a pixelláncok messzire elkalandozhatnak a számított élektől, vagy másképpen: a csúcspontok közé behúzott élek nem követik eléggé az eredeti vázat. Így ha a robot egyszerűen az élek mentén halad, akkor könnyen akadálynak ütközhet.

A probléma megoldására egy szegmentáló eljárással az élek finomabb felosz-

tását érdemes létrehozni, amely pontosabban követi az eredeti pixelláncot. Erre két különböző algoritmus létezik.

Wall és Danielsson az él és a lánc közötti területet határozzák meg ([193]). Az iteratív számítás egymásra támaszkodó háromszögek területének összegzését végzi. Ha a számítás eredménye egy küszöböt meghalad, akkor szükséges az él vágása.

Rosin és West algoritmusa az él és a lánc közötti maximális eltérést számítja ki ([149]). Az eljárás a legnagyobb eltérésnél vágja ketté az élt, és ezt rekurzív módon folytatja, amíg az új élhalmaz nem lesz elfogható approximáció (5.5. ábra).



5.5. ábra. Élek vágása Rosin és West algoritmusával. A vékony görbe vonal közelítése a szaggatott vonallal jelölt háromszögek és magasságvonaluk felhasználásával történik. Az eredmény a vastag vonalakból álló élhalmaz.

A két algoritmus összehasonlításaként Tombre és társai megállapítják, hogy Wall és Danielsson eljárása nagyon hatékonyan implementálható, de kevésbé pontos, mint Rosin és West módszere. Az utóbbihoz hozzátartozik, hogy kereszteződések közelében hajlamos sok kis él létrehozására ([187]).

Mivel a topológiai gráf készítésének célja a navigáció, ezért fontos, hogy a keletkező élek ne metsszék vagy közelítsék meg túlságosan az akadályokat és a falakat. Ezért a gráf lánchoz mért pontossága fontos szempont, így én Rosin és West algoritmusát implementáltam, melynek vázát az 5.3. algoritmus írja le. Az eljárás meghatározza egy lánc maximális távolságát a hozzá tartozó gráféltől. Ha ez az érték átlép egy küszöböt — esetünkben egy képpontot —, akkor a maximum helyén keletkezik egy új csúcspont két új éllel az eredeti végpontokhoz, és az algoritmus végrehajtódik a két új részre is.

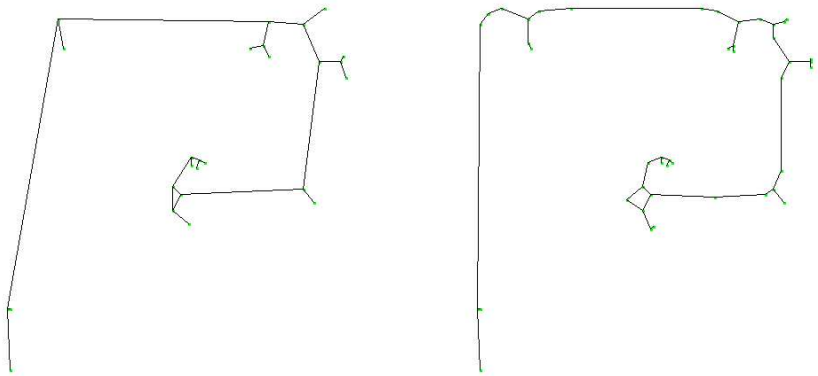
Az élek rekurzív vágása után az élek nyesése, azaz végük visszametszése is hasznos lehet, különösen a feltáratlan régiók közelében. Egyébként, ha a robot egyszerűen a gráf végpontjáig megy az ismeretlen terület közelében, akkor könnyen még nem érzékelt, de mégis jelenlévő falba ütközhet. Emiatt a tíz pixel-

5.3. algoritmus Az élek szegmentálása ([187] nyomán)

```
SZEGMENTÁLÁS(GRÁF,KEZDŐPONT,VÉGPONT)
   LÁNC := LÁNCMEGHATÁROZÁS(GRÁF,KEZDŐPONT,VÉGPONT)
   (MAXÉRTÉK,MAXPONT) := MAX_MAGASSÁG(LÁNC,KEZDŐPONT,VÉGPONT)
   IF MAXÉRTÉK > KÜSZÖB THEN
      GRÁF_ÉL_TÖRLÉS(GRÁF,KEZDŐPONT,VÉGPONT)
      GRÁF_ÚJ_CSÚCS(GRÁF,MAXPONT)
      GRÁF_ÚJ_ÉL(GRÁF,KEZDŐPONT,MAXPONT)
      GRÁF_ÚJ_ÉL(GRÁF,MAXPONT,VÉGPONT)
      SZEGMENTÁLÁS(GRÁF,KEZDŐPONT,MAXPONT)
      SZEGMENTÁLÁS(GRÁF,MAXPONT,VÉGPONT)
   END IF
END SZEGMENTÁLÁS
```

nél hosszabb élek hosszát négy pixellel csökkentettem, illetve az ennél rövidebb éleket négy hosszúságúra nyestem vissza.

Az 5.6. ábra jobb oldala mutatja a bal oldalon látott gráf optimalizált változatát a rekurzív vágás és a nyesés után. Ez az alakzat már a terep bejárható útvonalleírását adja.



(a) A láncolt gráf (b) Az optimalizált gráf

5.6. ábra. A gráf optimalizálása a váztól eltávolodó élek feldarabolásával és az utolsó élek visszanyesésével. Ennek következtében a gráfot követve a robot nem ütközik akadályba.

5.2.4. Útvonaltervezés topológiai gráffal

Amikor a feltérképezett és nem foglalt terület gráfja elkészült, a robotnak meg kell határoznia a felfedezés új irányát. Ehhez egy — a navigációról szóló fejezetben ismertetett — útvonaltérkép típusú tervezési stratégiát alkalmazok. A robot célja egyéb speciális feladat nélkül a terep teljes bejárása. Emiatt a gráf mindazon csúcsai célcsúcsnak számítanak, melyekhez közel található bejáratlan terület. Ennek meghatározásához a grafikából jól ismert Bresenham vonalrajzoló algoritmust használom[4] ([21]). Az eljárás sugárirányokban felméri, hogy fal vagy feltérképezetlen terület található-e a csúcs körül. Ha a 24 mérési irányból legalább kettő szomszédos ismeretlen részt jelez, akkor a csúcs felfedezésre alkalmasnak jelölhető.

A következő meglátogatandó célcsúcs meghatározásához az A^* algoritmust alkalmaztam ([59]). Ez a klasszikus eljárás a kezdőcsúcsból megtalálja a legrövidebb utat a felfedezésre váró célcsúcsok egyikébe. A kezdőcsúcs esetünkben a robot aktuális pozíciója. Az A^* algoritmus innen egy bejáratlan területhez közeli csúcsba vezet, ahol a további bolyongás hasznos lehet.

A legrövidebb utat, mint egy lista egymás utáni elemeit alkalmazva a robot pontosan a következő él mentén haladhat a következő csúcs felé, amíg a célcsúcsot el nem éri.

Az útvonaltervezés A^* algoritmusra alapozott megvalósulása mellett a már említett akadálykikerülő viselkedés is a rendszer részét képezi. Az alap mozgásmodul segítségével a robot nyílt terepen egyenesen halad, míg tárgyak közelében akadálykikerülő mozgásba kezd. Mivel a topológiai gráf létrehozása még mindig eléggé időigényes, ezért ez a művelet nem zajlik folyamatosan. Amikor az alap mozgásmodul használata során a robot nem talál elegendő új területet, vagy más szóval, amikor a feltérképezett rész nem nő eléggé — legalább száz képponttal száz lépésenként —, akkor jut szerephez a gráf alapú útvonaltervezés, mely generálja az időközben megváltozott környezet gráfját, és új utat talál rajta. Ez az alternáló működés a két eljárás előnyeit ötvözi, az üres tereket a robot egyszerű akadálykikerüléssel bejárja, majd a távolabbi térrészekbe az összetett navigáció viszi el.

[4] Az eredeti forráskód a http://en.wikipedia.org/wiki/Bresenham's_line_algorithm_C_code címen érhető el.

5.3. Eredmények

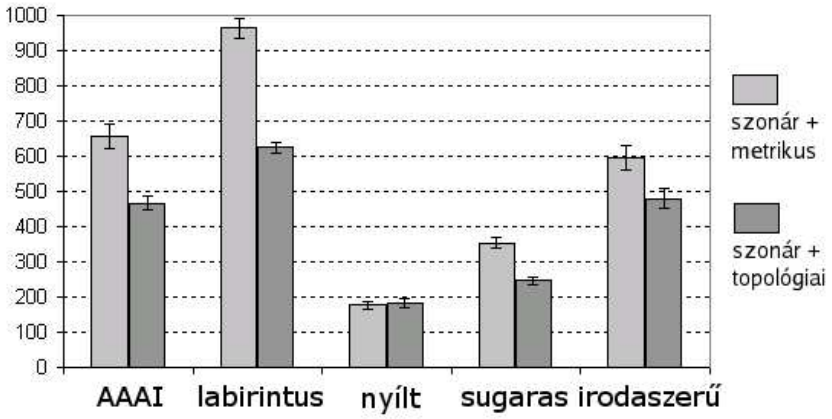
A kutatás során elkészítettem egy topológiai navigációs eljárást, amely foglaltsági hálóra alapozva működik a Webots szimulációs környezetben. Az értékiteráció felváltása topológiai gráffal a következő felfedezési irány meghatározásához hasznos módosításnak bizonyult. Egyrészről a megoldás közelebb áll az emberi kognitív térkép hierarchikus természetéhez, másrészről az új eljárás jobban teljesít.

Az új robotirányító program létrehozásához létező algoritmusokat használtam föl, melyeket a feladat feltételeihez igazítottam. Így a Tombre és társai által leírt láncoló algoritmust a csúcspontok eltérő definíciója miatt a helyes kezdeti gráf létrehozása érdekében módosítottam.

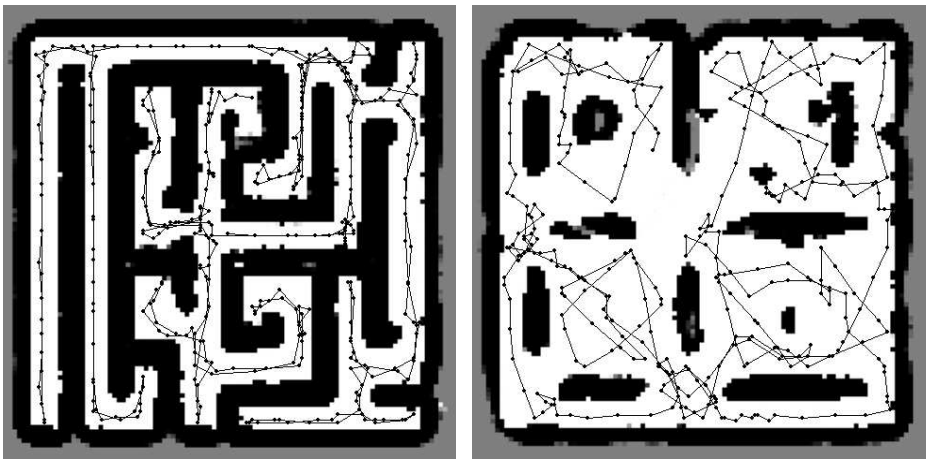
A létrehozott kontroller az előző fejezetnek megfelelően öt különböző környezetben, három kiindulási pozícióból, a robot mozgását meghatározó öt eltérő pszeudovéletlenszám-generátor kezdőértékkel indított futtatás során bizonyította képességeit. A robot sikerrel járta be a világokat, és elkészítette a topológiai gráfot a foglaltsági háló alapján. A futásokhoz szükségek idők terepenkénti átlagát az értékiterációs eredményekkel együtt az 5.7. ábra tartalmazza. Teljesítménymértékként továbbra is a 90%-os felderítéshez szükséges időt használtam.

A kis méretű nyílt terepet mindkét eljárás rövid idő alatt bejárja, lényeges időkülönbség nem tapasztalható. A kevés tereptárgy miatt az értékiteráció által kijelölt irányokban sem ütközik akadályokba a robot. A többi terepen azonban 65–80%-ra esik vissza a térképkészítés időszükséglete. Ez jórészt annak köszönhető, hogy a topológiai módszer esetében a navigáció a gráfon a robotot mindig felderítetlen térrészbe húzza, míg a költségmátrix alapú célkijelölés nem ennyire határozott, a döntés csupán lokálisan lesz optimális. Az új eljárásnál a robot a falak hozzávetőleges középvonalát jelentő gráféleken mozog, így kevesebb időt kell töltenie akadályok kikerülésével. Az értékiteráció a már nagyjából felderített terület néhány nem érzékelt részletéhez is visszatér, miközben a topológiai módszer csak akkor, ha a közelben van egy felderítetlenséget jelző gráfcsúcs.

A topológiai navigációs eljárást végző robot által létrehozott trajektóriákat az 5.8. ábra mutatja be egy-egy kiválasztott futás esetében a labirintusban és az irodaszerű környezetben. Az algoritmusok összevetésével látható, hogy a folyosókon a topológiai gráfot használva egyenesen halad a robot, ezáltal nem kell folytonosan apró irányváltásokat végeznie — ahogy az értékiteráció esetében —, és így



5.7. ábra. A terepek bejárásához szükséges idő másodpercben, ahogy az a valódi Khepera robotban mérhető lenne az értékiterációt és a topológiai gráfot használó algoritmusok esetén, világosszürkével az előbbi, középszürkével az utóbbi eljárást jelölve. Minden egyes oszlop egy környezetet jelent, és 15 futás átlagos idejét tartalmazza három különböző kiindulási pontból, 95%-os konfidenciaintervallummal.



(a) Labirintus (b) Irodaszerű környezet

5.8. ábra. A topológiai navigációt alkalmazó Khepera trajektóriája két terepen, az elkészült foglaltsági hálóra illesztve. A vonalon megjelenő pontok a robot helyét jelölik másodpercenként.

nem veszít időt. Az irodaszerű környezetben az is megfigyelhető, hogy a szobákat szisztematikusabban járja be a robot, mint a korábbi algoritmusnál. Ezúttal a teljes bejáráshoz szükséges minimális három helyett ötször lép át új szobába, szemben az értékiterációs eljárás kilencével. Ugyanakkor az is látható, hogy egy-egy szobán belül, illetve a találkozási pontoknál sok időt eltölt a robot. Ez abból adódik, hogy egy ismeretlen gráfcsúcshoz érve alapvető akadálykikerülésre vált át a robot mindaddig, míg elég nagy terepet sikerül így is földeríteni. Emiatt viszont előfordulhat, hogy az újonnan generált gráf egy felderítetlen csúcsához vissza kell térni. Ebből következően célszerűbb lenne a számításigényesebb gráf alapú útvonaltervezést tovább optimalizálni, és minél gyakrabban alkalmazni a fölösleges mozgások elkerüléséhez.

A két módszer közötti gyorsulás részben magyarázható a kezelendő objektumok számával. Az eltérés azt is megmutatja, hogy a környezet növekedésével milyen mértékben kerül hátrányba az értékiteráció a gráffal szemben a memóriaigény szempontjából. A 5.1. táblázat a különböző terepeken mért értékeket mutatja. Látható, hogy a gráf csúcsainak száma, amin a navigációs algoritmus működik, 20 és 120 között mozog. Ezzel szemben az értékiterációhoz tartozó pixelek száma nagyságrendekkel nagyobb, 11600 és 28900 közötti tartományban van.

5.1. táblázat. Kezelt objektumok száma

	Értékiteráció (Pixelek)	**Topológiai gráf** (csúcsok)	**Százalék**
AAAI verseny	23700	105	0.44
Nyílt	12800	50	0.39
Labirintus	28900	120	0.42
Sugaras	11600	20	0.17
Irodaszerű	28900	110	0.38

5.4. Kapcsolódó munkák

A kitüntetett helyek és a közöttük lévő relációk hangsúlyozása a fix celladekompozíción alapuló térképépítésnél később, a kilencvenes évek elején jelent meg. Kuipers és Byun az irányító és a geometriai réteg között helyezték el a topológi-

ait, és kellően robusztus irányítást felhasználva készítették el a középső szintet, amiből a metrikus információk leképeződtek ([92]). Tehát a térkép rétegeit fordított sorrendben hozták létre, mint ahogy azt a fejezetben bemutattam.

Mataric Toto nevű robotja iránytű és ultrahangos mérések alapján határozta meg a falak helyét, melyek mint tereptárgyak segítettek a kitüntetett helyek kiválasztásában. A gráf éleit az ezen pontok közötti közlekedési lehetőség adta meg ([109]).

A Franz és társai által használt Khepera robot egy panorámakamerát alkalmazott, aminek segítségével nyílt, falaktól távoli részekre is kimerészkedhetett a robot, a fontos helyek azonosítása könnyebbé vált. A körkörös kamera képeire alapozott nézetgráfon új csúcsot akkor hozott létre a robot, ha az aktuális látvány eléggé eltért a többi csúcsnál tárolttól. Gráfélek pedig akkor keletkeztek, ha az egyik nézetből a másikba közvetlenül tudott eljutni a robot ([58]).

Az előbbi két módszer közös tulajdonsága, hogy a topológiai gráfot közvetlenül az érzékelt környezet alapján hozzák létre, és nem használnak egy közbenső diszkrét reprezentációt, ellentétben az én munkámmal.

A topológiai térképépítés egy elterjedt formáját mutatja be Thrun, aki foglaltsági hálóra alapozva hozza létre a környezet gráfját ([182]). Szemben az általam alkalmazott vékonyításon alapuló vektorizációval, ő az üres térrész Voronoi-diagramját határozta meg. Ezek után a diagramon megkereste a falaktól lokálisan minimális távolságú kritikus pontokat, és kritikus éleket képzett belőlük, melyek a régiók határai lettek. A régiókkal izomorf alakzatként jött létre a topológiai gráf. A szerző az útvonaltervezést egy értékiterációs algoritmussal végezte, ami a régiók egy láncolatát határozta meg. A mozgáskivitelezéshez a lánc szomszédos hármasainak megfelelő régiók foglaltsági hálóján egy újabb, egylépéses értékiterációt hajtott végre. Az én eljárásomban erre a második értékiterációra nincs szükség, mivel az optimalizációval előálló gráf élei közvetlenül követhetők a cél irányába.

Valódi robot helyett szimulátorban végzett, topológiai gráfot létrehozó kísérletre példa Correll munkája, aki a Webots-ban foglaltsági háló készítése helyett lézeres mérések alapján egzakt celladekompozíciót végzett, és erre építve alakította ki a környezet gráfját ([39]). Módszerének további eltérése munkámhoz képest, hogy ő a térképmegosztás hatékonyságát vizsgálta az alkalmazott robotok számának függvényében, és nem navigációs eljárások teljesítményét hasonlította össze.

5.5. Következtetések

A fejezetben bemutattam eljárásomat, mely foglaltsági hálóra épít topológiai gráfot az útvonaltervezés elvégzéséhez. Bár ismert algoritmusok együttműködéséből született a megoldás, a teljes rendszer felépítése egyedinek tekinthető. A lényegesen jobb időeredmények igazolják ezt a megközelítést.

Egyrészről az eljárás kisebb memóriaigényűvé vált, hiszen a gráf csúcsainak száma két nagyságrenddel kisebb, mint a költségmátrix celláinak száma. Ez a terep növelésével egyre fontosabb lehet, komoly előny igazán nagy léptékű környezeteknél érzékelhető a költségmátrixszal szemben, amikor a bejárható terület az érzékelési horizontnál sokkal nagyobb.

A topológiai gráf másik előnye a futási idők csökkenése. Ez különösen a terep elnyújtott részein vagy nagy térrészek összekapcsolásakor jelent előnyt, amikor a gráf a teljes terep sajátságait tükrözi, szemben a költségmátrix lokálisabb döntéseivel, és így könnyebben segíti a felfedezésre váró területre a robotot. Ráadásul, mivel a gráfot követő robot a folyosók hozzávetőleges középvonalán halad, ezért a falak miatt is kevesebb irányváltoztatást kell végeznie, mint a költésmátrixot használó társának.

A topológiai gráf készítéséből adódó hátrány, hogy a szkeletonizáció eléggé zajérzékeny, emiatt a gráf élei az újrageneráláskor nem mindig esnek éppen ugyanoda, ami nem teljesen optimális döntéseket eredményezhet. Továbbá, mivel a gráf egyfajta tömörítést végez, ezért elképzelhető, hogy bizonyos részektől az élek mind távol vannak, így oda a robot csak az alapszintű felfedezés és akadálykikerülés során, esetlegesen jut el. Az előbbi problémán a gráf újrafelhasználásával, az utóbbin kamera bevezetésével lehetne segíteni.

6

Navigáció foglaltsági hálót bővítő kamerával

6.1. Bevezetés

A szem a legfontosabb érzékszervünk. Bár emberformájú gépek létrehozása nem feltétlen célunk, mégis természetes igény a látás mint érzékelési modalitás robotokba integrálása, hiszen a rendszer működésébe több bepillantást enged más érzékelőknél. Másrészről, ahogy az elfogadható képminőségű digitális kamerák ára esik, idővel ez a szenzor az egyik legolcsóbb típussá válik.

Ezen kívül, mivel az érzékelőknek megvannak a maguk korlátai és hibázástól sem mentesek ([174]), több szenzor használata — különösen eltérő érzékelési tartományokban — a begyűjtött információt pontosítja, megbízhatóságát növeli. Az új eszközök felhasználásából adódó előnyök kiaknázásához a különféle forrásból származó adatok integrálása szükséges ([155]). Ez a feladat jelenleg korántsem megoldott, a szenzorfúzió[1] a mobil robotika egyik legfontosabb nyitott kérdésköre.

Az előző fejezet az eredeti, értékiterációt használó navigációs algoritmus egy stratégiai szinten történő továbbfejlesztését mutatta be, az útvonaltervezéshez egy hatékonyabb eszközt adott. Emellett a navigáció taktikai részében is elképzelhető az eljárás bővítése. Ezen a szinten az egyik legfontosabb feladat az akadályérzékelés és -kikerülés. Ez a képesség a környező tárgyak, veszélyek érzékelését és

[1] sensor fusion

azok elkerülését foglalja magában ([155]).

Az akadályok távolságának észleléséhez sokféle szenzor használható. Ilyenek a már korábban bevezetett ultrahangos érzékelő, a szonár, valamint a lézeres távolságmérő, a radar és a doppler-radar is. Működési elvük szerint a kibocsátott jel a szenzor és az érzékelni kívánt visszaverő felület között megtett időt mérik. Ezen műszerek egy közös problémája, hogy a kis méretű, illetve lapos tárgyak érzékelése (pl. jég, olajfolt) nehézséget okoz.

Az akadályok távolságérzékelésének egy másik útját a látás biztosítja. A kamera a háromdimenziós világot leképezi a kétdimenziós képsíkba, mégis, több nézetet felhasználva, a kép valódi mélysége visszanyerhető. Ehhez a sztereó látás, az optikai folyam és a mélységi fókuszálás[2] képfeldolgozó eljárások használhatók.

Az akadályok érzékelésének egy másik, kevésbé elterjedt típusát a kinézetalapú módszer[3] jelenti ([53]). Ebben az esetben — a háromdimenziós világmodell létrehozása helyett — az algoritmus két dimenzióban működik, ami a számítási igényt csökkenti. Az akadályok a kép lokális tulajdonságai, a szín, a fényesség vagy a textúra alapján azonosíthatók. A tárgyak kikerüléséhez elég lehet a padlóhoz tartozó képpontok meghatározása az előbbi paraméterek alapján.

Kutatásom célja egy olyan robotirányító program létrehozása a Webots szimulációs környezetben, mely az eddigiektől eltérően az ultrahangos érzékelők mellett kamerát is használ a környezet feltérképezésére. A bővített Khepera robot feladata továbbra is a tesztkörnyezetek teljes bejárása.

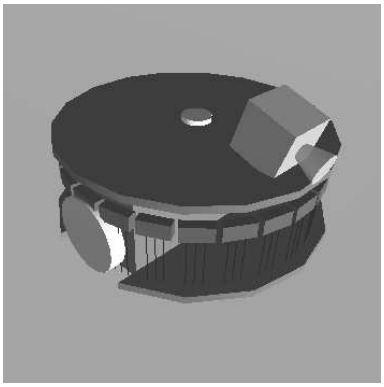
A feladat megoldásához kifejlesztettem egy kinézetalapú akadályérzékelő eljárást ([176]), mely a kamera képei alapján meghatározza az akadályok távolságát, és kiegészíti a foglaltsági hálót ezzel az információval. Ezt az új eljárást az értékiterációs és a topológiai navigációs programba integráltam, majd a két új algoritmus teljesítményét a korábbi, csak ultrahangos szenzort használó módszerrel hasonlítottam össze. Mivel a körkörösen elhelyezett szonároknak viszonylag kicsi a hatótávolsága, ezért joggal volt remélhető, hogy a robot tetejére helyezett távolabbra látó kamera jó hatással van a navigációra. Ezen kívül, mivel a szenzorok sohasem tökéletesek, ezért a belőlük kinyerhető információ egyesítése egy megbízható térkép készítéséhez szintén kihívást jelentett.

[2] stereo vision, optical flow, dept from focus

[3] appearance-based method

6.2. Szonár és kamera mérések egyesítése

A korábbi kísérletek folytatásaként a látás bevezetéséhez egy kamerát kapcsoltam a Khepera robothoz, majd az ultrahang alapú és a vizuális érzékelést egyaránt felhasználva hoztam létre a valószínűségi foglaltsági hálót. Ennek érdekében az eredeti robotra egy, a padló irányába 17 fokban megbillentett, 45 fokos látószögű, mindkét irányban 256 képpontos képet létrehozó kamera került (6.1. ábra).



6.1. ábra. A szimulált Khepera robot, jobb szélen az előre döntött fejkamerával.

A kamera bevezetésének lényege a szabad padló elhelyezkedésének meghatározása az akadályok távolságának megismeréséhez. A környező tárgyak azokban az irányokban vannak közel a robothoz, ahol a padló pixeljeinek sora csupán a kép egy kicsiny alsó részén látható, ezzel szemben a robot előtt nagy tér van, ha a pixelek magasra emelkednek. Vagyis a robot a fal és a padló pixeljeinek találkozási pontjából következtet a tárgyak távolságára.

Az eljárás alkalmazásához az alábbi feladatokat kell megoldani:

- a kamera képének feldolgozása
- az akadályok távolságának becslése
- a távolságok konvertálása foglaltságra
- útvonaltervezés

6.2.1. Képfeldolgozás

Az algoritmus első lépése a padlószínű képpontok kiválasztása a képen. Általában ez egy számításigényes eljárás, ami éldetektálást, szegmentálást, textúraanalízist

és jellemzőkiválasztást foglal magában. A munka jelenlegi fázisában a környezettel kapcsolatban feltételezek egy előre meghatározott padlószínt, sima talajt és a robot fölé benyúló akadályok hiányát. A tesztkörnyezetek ennek megfelelően viszonylag egyszerűek, mégis számos érdekes helyzetnek adhatnak teret. Mivel a terepek csupán tucatnyi textúramentes tárgyat tartalmaznak, így a feladat a padlószínű pixelek megkeresésére egyszerűsödik a különféle lehetséges megvilágítások figyelembevételével. Ezután egy bináris képet kell létrehozni, mely a padló színe alapján szeparálja a képpontokat. A szétválasztást három tényező határozza meg. Az aktuális képpont és a padló feltételezett színében lévő piros, zöld és kék komponensek arányai, a képpont színteltsége és egy előre megadott toleranciaszint. Ezen értékek együttesen meghatároznak egy küszöböt. A piros, a zöld és a kék színtartományokban az éppen vizsgált pixel és a padló színeltérésének pedig ez alatt kell lennie. Az eljárást a 6.1. algoritmus mutatja be, mely Michel korábbi munkája alapján készült ([42]).

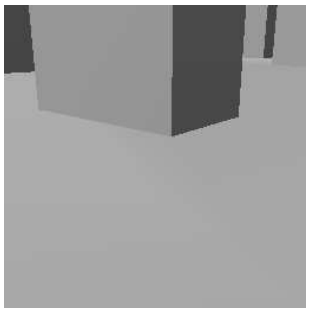
6.1. algoritmus Padlószínű képpontok meghatározása ([42] nyomán)

```
PADLÓPIXEL(TOLERANCIA,KÉPPONTSZÍN,PADLÓSZÍN,ELFOGADÁS)
   HASONLÓSÁG := SZÍNARÁNY(KÉPPONTSZÍN,PADLÓSZÍN)
   SZÍNESSÉG := MIN(100,ABS(KÉPPONTSZÍN.PIROS - KÉPPONTSZÍN.ZÖLD) +
                        ABS(KÉPPONTSZÍN.PIROS - KÉPPONTSZÍN.KÉK) +
                        ABS(KÉPPONTSZÍN.ZÖLD - KÉPPONTSZÍN.KÉK))
   KÜSZÖB := TOLERANCIA * ((2 - SZÍNESSÉG/100) + HASONLÓSÁG)
   ELFOGADÁS := ABS(KÉPPONTSZÍN.PIROS - PADLÓSZÍN.PIROS)<KÜSZÖB) AND
                ABS(KÉPPONTSZÍN.ZÖLD - PADLÓSZÍN.ZÖLD)<KÜSZÖB) AND
                ABS(KÉPPONTSZÍN.KÉK - PADLÓSZÍN.KÉK)<KÜSZÖB)
END PADLÓPIXEL
```

A 6.2. ábra egy a kamera által közvetített képet mutat, míg a 6.3. ábra a képfeldolgozás eredményét jeleníti meg.

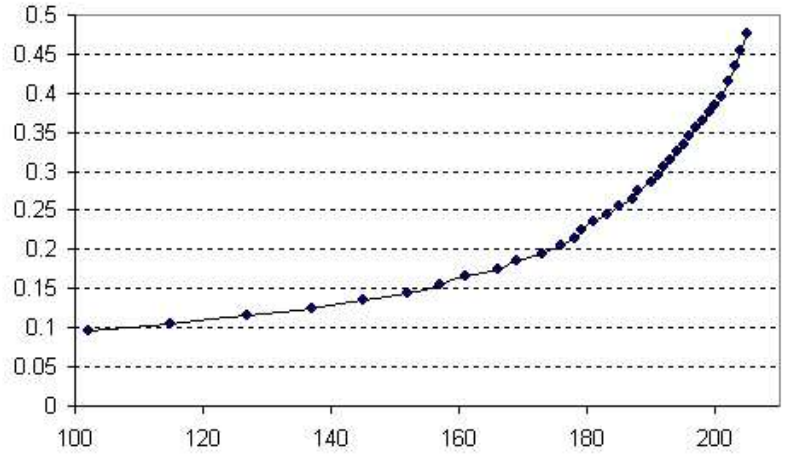
6.2.2. Távolságbecslés

Az akadályok távolságának meghatározása a kamera aljától számított padlószínű képpontok mennyiségén alapul. Előzetes próbafuttatások során készítettem egy leképező függvényt, mely a lehetséges értékekhez a mért fal- vagy akadálytávolságokat rendeli hozzá, ahogy azt a 6.4. ábra is mutatja. Az így generált távolságtáblázat kis számítási költséggel adja meg a hozzávetőleges tárgytávolságokat a 10 és 50 centiméteres tartományban.



6.2. ábra. A kamera képe

6.3. ábra. Az előfeldolgozott bináris kép. A padlószínű képpontok világossal vannak kiemelve.



6.4. ábra. Tapasztalati úton meghatározott összefüggés, mely a padlószínű képpontok függvényében megadja a tárgyak távolságát. A vízszintes tengely a kamera aljától számított megszakítás nélküli padlószínű képpontok számát jelöli a 256 pixel magas kamerán, míg a függőleges tengely az adott irányban becsült tárgytávolságot jelenti méterben. A 100 képponttal jelölt minimális távolságnál közelebb a szonár az elsődleges információforrás, a maximális távolságnál, vagyis 205 képpontnál, a becslés pedig annyira pontatlanná válik, hogy nem érdemes a mért értékekből foglaltságra következtetni.

6.2.3. Távolság leképezése foglaltságra

Az algoritmus következő lépése a távolságok foglaltsági hálóra képezése. Ez az érzetek egyesítésének, a szenzorfúziónak az ideje. A megoldás egyszerűsége a foglaltsági háló népszerűségének egyik fő oka. Miután az eljárás ugyanúgy távolságadatokat kap, mint az első navigációs algoritmusnál leírt szonáradatok feldol-

gozásakor, ezért a számítási mód is a korábbival egyező: az érzetekből kinyert, adott pozícióra vonatkozó valószínűségek inkrementális módon adhatók hozzá az eddigi valószínűséghez. Ezúttal azokat a cellákat kell meghatározni, melyek a robot látószögébe és távolságába esnek, ami az ultrahangos mérésekhez képest egy nagyobb környezetet jelent. Ezekhez a pontokhoz nagy foglaltsági valószínűség rendelhető, ha a kamera akadályt jelez, és kis valószínűség a robothoz közelebb.

6.2.4. Útvonaltervezés

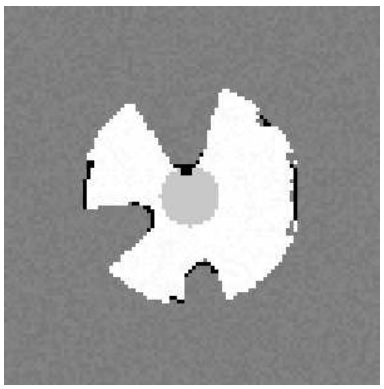
A bővített foglaltsági háló ismeretében a navigáció a már ismert ultrahangos akadálykikerülés, a topológiai gráfot használó A^* algoritmus, valamint a fényképezés kombinációja. Bár a képfeldolgozás viszonylag egyszerű folyamat, mégsem működhet folytonosan, mivel időigényes. A robot, amikor épp nem a topológiai gráfot használja, ötven lépésenként ellenőrzi, hogy érdemes-e fényképet készíteni. Ez a robottól a különböző irányokban közvetlenül érzékelhető feltérképezetlen terület nagyságától függ. Vagyis egy olyan csomóponthoz érve, ahol a keresztező irányok még nem bejártak, célszerű körbefordulni. A körbefordulás hasznát az előző fejezetben leírt Bresenham-algoritmuson alapuló eljárás határozza meg, ezúttal a robot aktuális pozíciójában alkalmazva ([21]). Ha érdemes körbefordulni, akkor a robot húsz fokonként fényképet készít, melynek eredményét a foglaltsági hálóba integrálja. Ezen kívül induláskor is a körbefordulás az első tevékenység a környék kezdeti feltérképezéséhez. A 6.5. ábra egy ilyen kezdeti körbefordulás eredményét mutatja a nyílt terepen.

A korábbi eljárások egy másik lehetséges bővítése a kamera használata az értékiterációt alkalmazó algoritmusban. Ennek bevezetésével a kezdeti metrikus navigáció két kiterjesztésének, a topológiai gráfnak és a kamerának a szerepe különválasztható, az eredmények önállóan is értelmezhetőkké válnak.

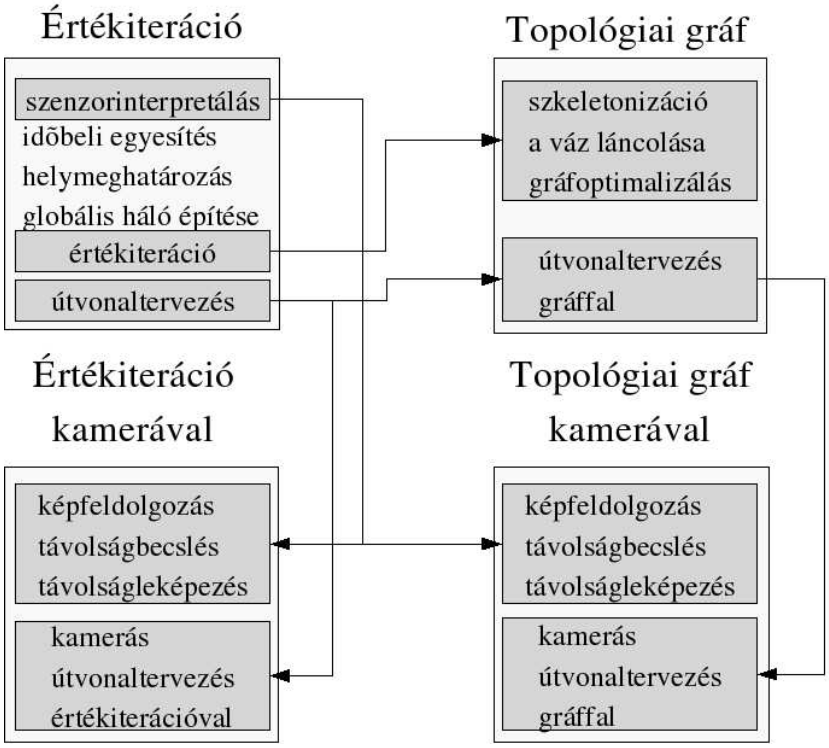
6.3. Eredmények

A kutatás során a korábbi navigációs eljárások két kiterjesztését hoztam létre a Webots szimulációs környezetben. Bevezettem a kamera képének felhasználását mind a metrikus, mind a topológiai navigációban. A 6.6. ábra az eljárások közötti összefüggéseket mutatja.

A kialakított robotirányító programokat a már ismert öt különböző kísérleti

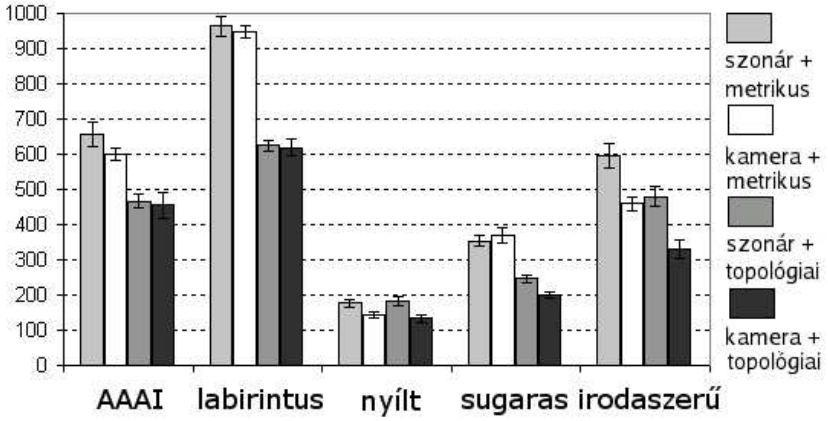


6.5. ábra. A kiterjesztett foglaltsági háló a körbefordulás után. A világos térrész a néhány környező fekete színű akadállyal a kamera hatótávolságát mutatja az alkalmazott kinézetalapú eljárás esetén. A középső kisebb szürke terület a szonárok érzékelési távolságát jelzi.



6.6. ábra. Az algoritmusok kapcsolata. A nyilak a leszármazás irányát jelzik. A szürkével keretezett részek módosultak, illetve bővültek, míg a többi elem változatlan maradt.

terepen teszteltem, három különböző kiinduló pontból, öt eltérő pszeudovéletlenszám-generátor kezdőértékkel meghatározva a robot mozgását. Teljesítménymértékként ismét a 90%-os felderítéshez szükséges időt használtam.

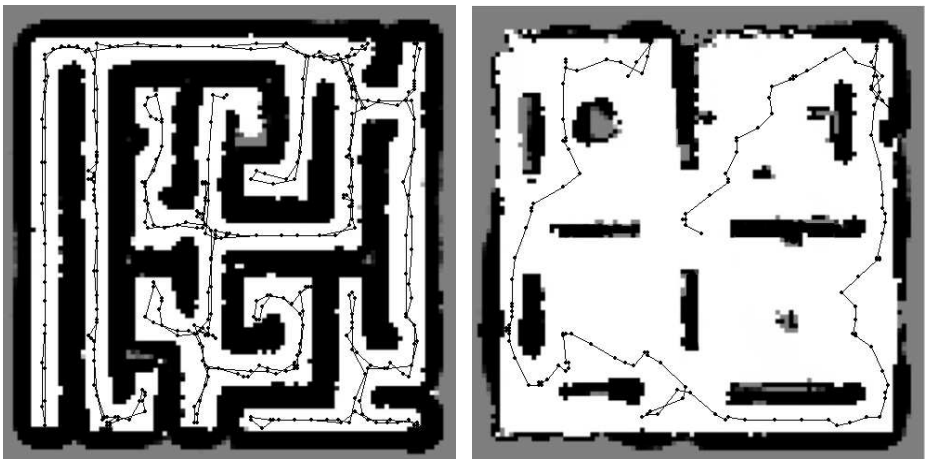


6.7. ábra. A terepek bejárásához szükséges idő másodpercben, ahogy az a valódi Khepera robotban mérthető lenne a négy vizsgált algoritmus esetén. Az értékiterációt szonárral világosszürke, az értékiterációt kamerával fehér, a topológiai gráfot szonárral középszürke, míg a topológiai gráfot kamerával fekete szín jelzi. Minden egyes oszlop egy környezetet jelöl, és 15 futás átlagos idejét tartalmazza három különböző kiindulási pontból, 95%-os konfidenciaintervallummal.

A kísérlet eredményeit a 6.7. ábra összegzi, a korábbi tesztek tükrében. Kamera használata az érzékelésben a szonárok kiegészítéseként a topológiai gráf létrehozásához hasznos bővítésnek bizonyult. Az eljárás a labirintus és az AAAI verseny terepek esetén az előző kiterjesztéssel nagyjából megegyező módon teljesített, míg a nyílt, az irodaszerű és a sugaras terepnél az időszükséglet 70–80%-os. Az utóbbi három környezetben az eredmények közötti eltérés a nagy nyitott terekkel magyarázható, ahol is a kamera egyszerre a terep tekintélyes részét tudja felmérni. Az első két kísérletben ugyanakkor a keskeny folyosók és a kis szobák vannak túlsúlyban. Ezeknél az akadályok érzékelése a padló képe alapján nem jelent lényegi előrelépést a szonárhoz képest, hisz a szobák esetén ez utóbbi is elegendő, folyosóknál viszont kamerát használva is el kell jutni az átellenes végig.

Ezzel szemben az értékiterációt kamerával bővítve az időeredmények nem javulnak lényegesen. A nyitott térrészen a feltérképezett terület aránya gyorsan nő, ami a nyílt terep és az irodaszerű környezet esetén a bejárás végén is előnyt je-

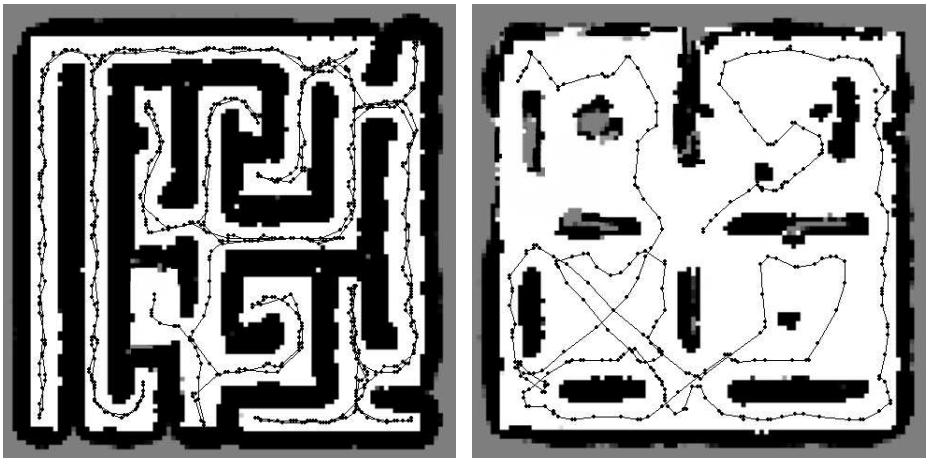
lent. Ugyanakkor a robot továbbra is a lokális környezetbe gyűjtött információk alapján dönt, ami nem feltétlenül optimális, a körbefordulások többletideje ezért hosszú folyosókon, kis szobákban nem térül meg a topológiai gráf kamerás kiterjesztésénél leírtakhoz hasonlóan.



(a) Labirintus (b) Irodaszerű környezet

6.8. ábra. A Khepera trajektóriája két terepen, az elkészült foglaltsági hálóra illesztve a kamerát a topológiai navigációval használva. A vonalon megjelenő pontok a robot helyét jelölik másodpercenként.

A kamerával felszerelt topológiai navigációs eljárást végző robot által létrehozott trajektóriákat a 6.8. ábra bal és jobb oldala mutatja be egy-egy kiválasztott futás esetében a labirintusban és az irodaszerű környezetben. Az előbbi környezetben a robot továbbra is kötött útvonalon haladhat, a fényképezés csak akkor jelent előnyt, ha épp keresztutaknál történik, vagy a bejárás végeztével, amikor az utolsó szobába nem kell bemenni, ahogy az a kép felső harmadának közepén látható. Az irodaszerű terep esetén a robot lényegesen rövidebb utat tesz meg, mint a korábbi két algoritmusnál, emiatt a futási idő is jobb, mint azoknál. Bizonyos szobákba be sem megy, csupán a bejáratuknál fényképez a robot. Ilyen eset a kép alsó részén látható: a robot elindul a terep szélén lévő folyosó felé, a bejáratánál a teljes hosszát lefényképezi, majd a nagyobb szoba felé folytatja útját. Ugyanakkor a térkép megbízhatósága is láthatóan csökkent, mivel bizonyos részeken — így a bal felső sarokban — csupán egy-egy, esetleg kevésbé jól sikerült irányból készült fénykép alapján következtet a foglaltságra az eljárás, a pontosabb



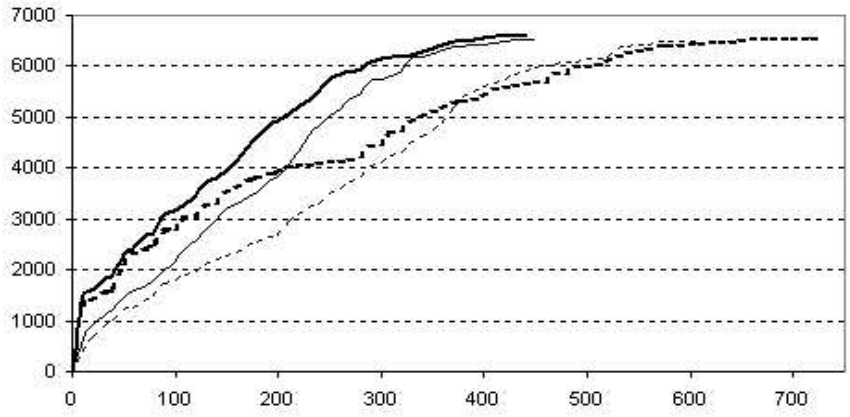
(a) Labirintus (b) Irodaszerű környezet

6.9. ábra. A Khepera trajektóriája két terepen az elkészült foglaltsági hálóra illesztve a kamerát az értékiterációs navigációval használva. A pontok a robot helyét jelölik másodpercenként.

valószínűségekhez több mérés összegzésére volna szükség.

A 6.9. ábra a kamerával felszerelt értékiterációs eljárást végző robot által létrehozott trajektóriákat mutatja be egy-egy kiválasztott futás esetében a fenti két terepen. A folyosókon tapasztalható, falakat is érintő, irányváltoztató mozgás itt is látható, csakúgy, mint a legkorábbi eljárás esetében. A labirintusban az algoritmus a folyosókon is körbefordulva fényképez, ami további lassulást eredményez. Az irodaszerű terep bejárása az előző, topológiai navigációhoz hasonlít a gyakori fényképezéssel, ami a szobákon való gyors áthaladást segíti, de egyúttal a térkép megbízhatóságát is csökkenti.

A kamera használatának a gyorsabb bejárás melletti másik előnye a topológiai gráf használatakor, hogy a feltárt terület nagysága a futás során mindig magasabb, mint a csak szonárt alkalmazó eljárások esetében. Ez a nyitottabb környezetekben egyértelmű, de az összes terepen észlelhető. A 6.10. ábra az AAAI verseny környezetében végrehajtott öt futás átlagát mutatja. Az előbbi megfigyelésen túlmenően az is látható, hogy a kamerát használó értékiterációs eljárás a szonár alapú topológiai gráfos algoritmusnál is jobban kezd, de nagyjából 200 lépés után a tendencia megfordul, és a jobb útvonaltervező eljárás az előbbi fölébe kerekedik.



6.10. ábra. A feltárt terület nagyságának alakulása a különböző algoritmusokban az AAAI versenykörnyezetben. A vízszintes tengely az időt, a függőleges a foglaltsági háló módosított celláinak számát mutatja. A vastag, illetve vékony vonalak öt futás átlagát jelentik kamerával és anélkül, ebben a sorrendben. Az értékiterációs teszteket szaggatott, míg a topológiai gráfos kísérleteket folytonos vonalak jelzik.

6.4. Kapcsolódó munkák

A távolságalapú akadályérzékeléssel foglalkozó cikkek nagy számával ellentétben a kinézetalapú módszer csupán az utóbbi időben indult fejlődésnek. Mégis, érdekes módon, az algoritmus használata egészen a kezdetekig visszavezethető: Shakey, az első autonóm robot a padlósík képét dolgozta fel egy monokróm kamerát használva ([126]). Az eljárás során a képen éleket határozott meg, melyből a legalsó jelentette számára az adott irányban legközelebb lévő akadályt. A fehér padló nem tartalmazott textúrát, míg a fal alsó részét feketére festették a kontraszt növelése érdekében.

Horswill éldetektáló algoritmusát szintén valódi robotba helyezte, mely a mintázat nélküli padlószőnyeg képpontjait keresi a képen. A robot elfordul azokból az irányokból, amelyekben a képen éleket érzékel. Ez a megoldás a szerző állítása szerint sokkal megbízhatóbb, mint az általa vizsgált szonáralapú eredmények ([75]).

Lorigo és társainak algoritmusa a Mars köves felszínére hasonlító, strukturálatlan környezetben működik ([101]). Az akadályok érzékelésére a kis felbontású

képek RGB, HSV[4] és fényesség információt tartalmazó nézetei szolgálnak. A szerzők az üres tér színét megadó referenciaként a kép bal alsó sarkát használják, feltételezve, hogy ott nincs akadály. A képek párhuzamos feldolgozása a három tartományban az algoritmust meglehetősen robusztussá teszi a változásokkal szemben, de sajnos a képmezőn kívüli tárgyak elakadást okozhatnak.

Martin doktori dolgozatában LISP nyelvű, képfeldolgozó operátorokból álló genetikus programok segítségével határozza meg a falak és az ajtók távolságát ([105]). A kísérletek előtt Lorigo módszerével előfeldolgozott képek segítenek a populáció egyedeinek kiértékelésében. A példák kézi módosításával a hibák száma mintegy hatodára csökkent.

Ulrich és társai szintén referencia-hisztogramokat használnak az üres helyek meghatározására, de ők azt feltételezik, hogy az a térrész üres, melyen a robot éppen áthaladt, vagyis a korábbi képek alapján döntenek, így kerülve el a Lorigo módszerében meglévő komoly előfeltételt ([192]). Az eljárás sokféle külső és belső környezethez alkalmazható, felhasználói beavatkozásra csak a padló felülethatárainál van szükség, ahol a robotot át kell segíteni az új területre, hogy betanulja annak kinézetét.

A kinézetalapú akadályérzékelés módszerének alkalmazhatóságát Hoffmann és társai munkája is igazolja, ők sikeresen oldották meg a feladatot Aibo robotokkal a 2004-es Robocup versenyen, hiszen győztek az akadálykikerülési szekció dinamikus környezeteiben ([72]).

A bemutatott eredmények közös tulajdonsága, hogy a komolyabb képfeldolgozási eljárásokat valódi robotban valósítják meg, szemben az én szimulátorra épülő munkámmal. Ezen kívül én a kinézetalapú akadályérzékelést használó eljárást más navigációs algoritmusokkal vetem össze. Szimulátorban végzett kísérletre példa Stepan és társai munkája, akik többek között a Webots-ban hoztak létre foglaltsági hálót egy kamera képe alapján, de ők nem kinézet-, hanem az elterjedtebb távolságalapú akadályérzékelést valósították meg ([163]).

6.5. Következtetések

A fejezet bemutatta, hogy az eddig szonárt használó környezeti leképezés miként bővíthető vizuális információval.

[4]RGB – piros, zöld, kék, illetve HSV – színárnyalat, telítettség, világosság

A foglaltsági hálón alapuló, topológiai gráfot használó navigáció kiterjesztése egy kamera képének kezelésével előnyöket és hátrányokat is jelent. Bár a kamera képének felhasználásához mindig hozzátartozik valamilyen előfeltételezés a környezetről ([75]), a padló színének rögzítése, más hasonló színű alapzattal rendelkező tárgyak kizárása komoly megszorításokat jelent. Az eljárás nem eléggé robusztus, nagyobb fény-árnyék változások elronthatják az eredményt. Ezen kívül bizonyos esetekben kimaradhatnak olyan szobák a térképezésből, melyek hosszú folyosókról nyílnak, és ajtajukat a robot messziről fényképezve nem veszi észre.

A kamera bevezetése ugyanakkor számtalan új lehetőséget nyit a mobil robotok előtt. Az egyszerűbb érzékelők puszta távolságinformációjával szemben a környezet egyéb jellegzetességei, úgymint színek, mintázatok, feliratok vagy a tükröződés olyan információkat adhatnak, melyek egyébként nem állnak rendelkezésre. A kamera képének felhasználása kitágítja a robot körül érzékelt teret, és a több elvégzendő számítás ellenére is gyorsabb bejárást biztosít. Az eredmények azt mutatják, hogy a kamera nagyobb üres terek esetében előnyös igazán, amikor egy körbefordulással komolyabb térrészt érzékel a robot, mint a nyílt terepen vagy az irodaszerű környezetben. A bejárási idő 70%-ra csökkent ezekben az esetekben. Amikor kis szobák és keskeny folyosók dominálnak, mint például a labirintusban, akkor a kamera szerepe sokkal kisebb.

Az értékiteráció kiterjesztése kamerával nem igazán javít a futási időkön. Tehát jobb érzékelők alkalmazása nem feltétlenül jelent sikeresebb térképépítést, ehhez megfelelő útvonaltervezési eljárás is szükséges.

6.6. Folytatási irányok

Bár a Webots szimulátorban végzett navigációs kísérletek eredményei bíztatóak, számos további kérdést vetnek föl, ezért célszerű néhány további feladat elvégzése.

A teljesen autonóm robot létrehozásához érdemes volna egy olyan lokalizációs eljárást felhasználni, melyhez nem kötődik külső segítség, hanem a robot — például odometrikus információk és a kamera alkalmazásával — elfogadható pontossággal megismerné koordinátáit. Ezen kívül a topológiai gráfot a foglaltsági hálótól függetlenül is el lehetne készíteni. Ez többek között kisebb memóriaigényt jelentene, másrészt a tisztán topológiai és a hierarchikus térkép összehasonlításá-

nak lehetőségét nyújtaná.

A kinézetalapú világmodellezésnél a padló képének meghatározását érdemes volna robusztusabbá tenni. Változó fényviszonyokra, mintázatokra, színekre való érzéketlenséggel az eljárás sokat javulhatna. Továbbá, előre meghatározott padlószín alkalmazása helyett, rugalmasan alkalmazkodva az aktuális környezethez, a robot használhatóbbá válna. Az algoritmus további hasznos kiterjesztése lehet egy panorámakamera, amivel a robot körbefordulásához szükséges idő megtakarítható.

Hasznos volna a kísérletek eredményeit más, az eddigiektől akár lényegesen eltérő navigációs algoritmusokkal összevetni. Az algoritmusok valódi robotba ágyazása pedig az ismertetett megoldások további igazolása lehetne. Végül a robotot magas szintű feladatok elvégzésére kellene használni, úgymint porszívózás, fűnyírás, hisz jórészt emiatt építünk intelligens gépeket: hogy helyettünk dolgozzanak.

7

Hangyák ételgyűjtése formális szemmel

7.1. Bevezetés

Tegyük föl, hogy a következő logisztikai problémánk van! Rendelkezésünkre áll mobil ágensek, nevezzük nevén, robotok egy csoportja, melyek tárgyakat tudnak szállítani egyik helyről a másikra. A robotok viselkedése az alábbiakkal foglalható össze:

- egy központi helyről indulnak és felfedezik a környezetüket,
- azokat a helyeket igyekszenek megtalálni, ahol nagy mennyiségben találhatók érdekes tárgyak,
- ilyen helyek megtalálásakor egyenként a központba szállítják a tárgyakat,
- a kitüntetett helyeket olyan gyakran próbálják meglátogatni, amilyen gyakran csak lehet, hogy minél többet tudjanak begyűjteni az ott lévő tárgyakból.

Az ilyen típusú feladat gyűjtögetés néven jól ismert a kollaboratív és kollektív robotika művelői számára, ahogy azt Balch doktori dolgozatában bemutatja ([7]). Mégis olyan kérdéseket vet föl, melyek nem könnyen válaszolhatók meg. Amikor egy különleges helyet megtalál egy robot, ezt az információt miként kell átadni a társaknak? A szerencsés első szétsugározhat egy üzenetet, de mi legyen abban? A pontos robotkoordinátákat nehéz meghatározni, ez egy bonyolult lokalizációs feladat ([183]). Továbbá a koordináták ismeretében egy oda vezető út még nem magától értetődő, a megoldáshoz útvonaltervezés szükséges. A környezet egyéb jellegzetességei (pl. környező tereptárgyak képe) nehezen átadhatók és feladat-

függők lehetnek.

Következő kérdés, hogy miként kell a robotoknak eloszlaniuk a párhuzamosan megtalált érdekes helyek között? A helyek népszerűségének arányban kell állnia megközelíthetőségükkel és gazdagságukkal. Ezen a ponton a robotoknak egy összetett egyeztetési eljárást kell végezniük, hogy az erőforrások kiaknázását maximalizálják, ugyanakkor teret hagyjanak egy kevés explorációnak is ([8]).

Az eddigieken túlmenően a robotoknak a feladat dinamikus jellegével is meg kell birkózniuk, azaz források kimerüléséhez, újak felbukkanásához és az ismert forrásokhoz vezető utak megváltozásához is alkalmazkodniuk kell.

A vizsgálat során figyelmen kívül hagyjuk, de valójában lényeges kérdés a robotok ütközésének kérdése, hiszen ha túl nagy számban vannak jelen, akkor egymást akadályozzák a munkavégzésben, ahogy arra Goldberg és Mataric ([62]), valamint Holland és Melhuish munkája is példa ([73]).

A vázolt logisztikai problémára adott hatásos válasz a természetben megtalálható: rajban élő biológiai ágensek, azaz például hangyák hatékonyan oldják meg a gyűjtögetés feladatát. A feladatra ők a *stigmergia*[1] módszerét alkalmazzák, mely a környezeten keresztüli kommunikáció segítségével szabályozza a munkafolyamatot és motiválja az élőlényeket.

A fejezetben a stigmergikus ételgyűjtés rövid ismertetése után bemutatok egy formális modellt, melynek keretein belül megvizsgálom, hogy mi a kapcsolat a környezet kezdeti rendezetlensége és a hangyák ételgyűjtési teljesítménye között ([66], [67]). Ezen kívül azt is tanulmányozom, hogy a feladat konfigurációja miként befolyásolja a hangyák viselkedését, a köztük kialakuló koordinációt. A fejezetben ismertetett eredmények Gulyás Lászlóval és Laufer Lászlóval közös munka kapcsán jöttek létre.

A motivációt jól foglalja össze Parunak és Brueckner gondolata: „Autonóm folyamatok fejlődése a természetben a rendezetlenség és nem a szervezettség irányába mutat. (...) Csak akkor lehetünk sikeresek ágens alapú rendszerek létrehozásában, ha megértjük rendezettség és rendezetlenség összefüggéseit.” ([136]) Bizonyos értelemben Simon híres hangyáját vizsgáljuk ([157]). A szerző szerint a homokban mászó hangyának bonyolult működést lehet tulajdonítani, miközben csupán a felszín egyenetlenségeit követi. Ugyanígy az egyszerű Braitenberg-járművek követik az érzékelt környezet finom változásait, és bonyolult trajektóriát

[1] A fogalmat Grassé 1959 alkotta meg a *stigma* (jel) és a *ergon* (munka) görög szavak felhasználásával ([63]).

mutatnak be. Ehhez hasonlóan az ételgyűjtési feladatban meglévő bonyolultságot és a feladatot megoldó hangyák viselkedésében meglévő bonyolultságot próbálom összevetni.

7.2. Ételgyűjtő hangyák

Amikor a hangyák kirajzanak a fészkükből, akkor egy darabig úgy tűnik, céltalanul mozognak, de hamarosan egy határozott nyomot képeznek a fészek és az ételforrás között (7.1. ábra). Meglepő módon a nyom nagyjából egybeesik az optimális útvonallal. Ha több ételforrás is van a közelben, akkor távolságuk alapján szüretelik le azokat.

7.1. ábra. Ételgyűjő hangyák ösvényt képeznek a fészek és az ételforrás között. Az útvonal azért válik láthatóvá, mert megtisztították az akadályoktól.

Minden egyes hangya véletlen irányba indul el a fészekből egy fészekferomonnyomot[2] hagyva maga után. Amikor egy hangya ételt talál, akkor hazafelé indul el, miközben ételferomonnyomot hagy hátra. Az ételkereső hangyák az étel feromonjának, míg a hazafelé tartó hangyák a fészek feromonjának magasabb koncentrációja felé irányítják lépteiket. Egy bizonyos feromonszint alatt, illetve egy meghatározott valószínűségnél a hangyák véletlen módon mozognak. Ez az utóbbi komponens segít az exploráció-exploitáció közötti egyensúly megteremtésében, ami a viselkedésoptimalizáció lényeges eleme ([166]).

[2] A feromon egy illékony anyag, mely kibocsátása környezetében szétterjed, és a forrásától távolodva egyre csökkenő koncentrációt mutat.

Az ételgyűjtés a hangyatársadalom egésze szintjén egy összetett, szervezett viselkedés, miközben a hangyaegyedek meglehetősen egyszerű, valószínűségi szabályhalmazt alkalmaznak. Pontosan ez az ágensszintű egyszerűség az, ami különösen vonzóvá teszi az eljárást a komplex osztott rendszerekkel kapcsolatos problémák vizsgálatakor. A kolónia szintjén megjelenő szervezett viselkedés kulcsa a *környezet általi kommunikáció*. A hangyák mindig lokális módon kommunikálnak (a feromont aktuális helyükön kibocsátva), de a fizikai környezet ezt az információt elterjeszti a diffúzió és az evaporáció segítségével. Minden globális kommunikáció nélkül valósul meg, a hangyaboly makroszintű céljait a hangyák mikroszintű viselkedésének koordinálásával és finomhangolásával éri el. A környezeten keresztüli kommunikáció szabályozó szerepét az ételgyűjtés folyamatában a 7.2. ábra szemlélteti.

7.2. ábra. Hangyák ételgyűjtése a stigmergia révén

7.3. Az ételgyűjtés egy formális modellje

A fejezet hátralevő részében a fent leírt viselkedés Deneubourg és társai által kidolgozott modelljét használom ([46]). Adott N hangya (1-től N-ig indexelve), melyek egy diszkrét, kétdimenziós, S méretű, L-lel jelölt periodikus világban, azaz egy tóruszon élnek. Jelölje $l_i^t \in L$ az i. hangya helyét a t pillanatban és $f(p) \geq 0$ az étel mennyiségét (diszkrét módon) $p \in L$ pozícióban. Kezdetben minden hangya a fészekből indul, azaz $l_i^0 \in K \; \forall i \in [1..N]$. A fészek egy R sugarú kör, mely L tetszőleges pontján helyezkedhet el. (A periodikus peremfeltétel miatt K tekinthető a terep középpontjának.) A hangyák feladata az összes étel fészekbe gyűjtése. A későbbiekre tekintettel F jelöli az összeszedendő étel mennyiségét: $F = \sum_{p \in L} f(p)$.

A hangyák nyomot hagynak, amikor aktuális pozíciójukban feromont helyeznek el. A feromon típusa függ attól, hogy a hangya éppen hazafelé tart, vagy ételt keres, míg a feromon mennyisége (A) függ attól a T időtől, amit a hangya az aktuális tevékenységgel megszakítás nélkül töltött: $A = max(m - (T-1) \cdot d, 0)$, ahol m és d a modell paraméterei. A kibocsátott feromon diffundál a szomszédos mezőkre, és lassan el is párolog. Így $\phi_z^t(p)$ adja meg a z típusú feromon mennyiségét t időpillanatban p helyen.

$$\phi_z^t(p) = \left(\rho \cdot \phi_z^{t-1}(p) + (1-\rho) \cdot \frac{\sum_{q \in L, |p-q|=1} \phi_z^{t-1}(q)}{8}\right) \cdot (1-\delta) + \sum_{i \in [1,N], l_i^t = p} A_i^t$$

ahol δ a párolgás, ρ a diffúzió sebessége, mindkettő a modell paramétere. A_i^t az i. hangya által t időpontban kibocsátott feromon mennyisége.

Járkálásuk közben a hangyák egyszerű valószínűségi szabályt alkalmaznak: a legmagasabb feromonszintű szomszédos mezőre mozdulnak el (a céltól függően hol fészekferomont, hol ételferomont követve). Egy bizonyos küszöb alatt és egy megadott w valószínűséggel véletlenszerűen mozognak. Egyúttal a hangyák igyekeznek egyenes vonalban mozogni, lehetőleg elkerülve a fordulásokat. Ennek megfelelően h_i^t jelöli az i. hangya irányát t időpontban, mely egy l_i^t-vel szomszédos mező. Ezen kívül $bal(h)$ és $jobb(h)$ jelöli a h-tól balra és jobbra lévő irányokat. A hangya mozgása az alábbiaknak felel meg:

$$l_i^{t+1} = \begin{cases} véletlen(h_i^t) & w \text{ valószínűséggel,} \\ feromon\_keresés(h_i^t) & 1-w \text{ valószínűséggel} \end{cases}$$

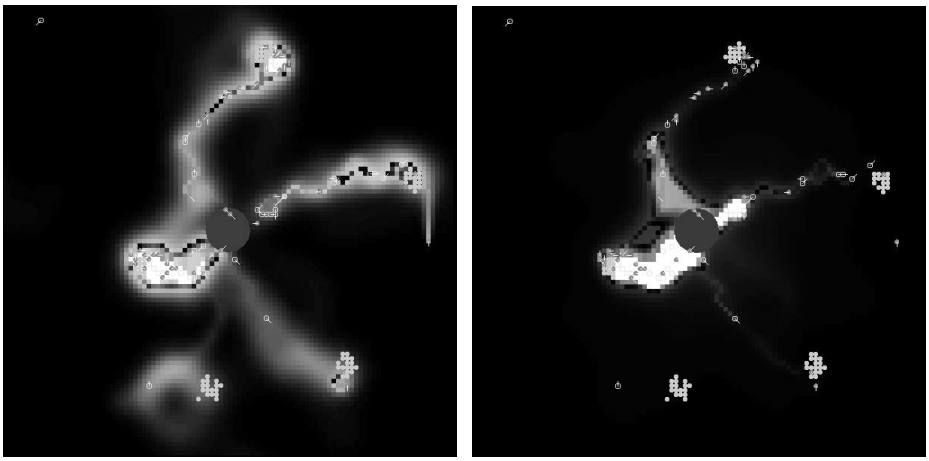
ahol $véletlen()$ és $feromon\_keresés()$ definíciója az alábbi:

$$véletlen(h_i^t) = \begin{cases} h_i^t & 2/3 \text{ valószínűséggel,} \\ bal(h_i^t) & 1/6 \text{ valószínűséggel,} \\ jobb(h_i^t) & 1/6 \text{ valószínűséggel} \end{cases}$$

$$feromon\_keresés(h_i^t) = \begin{cases} véletlen(h_i^t) & \text{ha } max\{\phi_z^t(D)\} < \alpha, \\ argmax\{\phi_z^t(D)\} & \text{egyébként} \end{cases}$$

ahol $D = \{h_i^t, bal(h_i^t), jobb(h_i^t)\}$ és α a modell paramétere.

A szimuláció egy állapotát mutatja a 7.3. ábra két része, külön ábrázolva az étel- és a fészekferomont.



7.3. ábra. Ételgyűjtés a szimulátorban. A szürke árnyalatok bal oldalon az ételferomont, jobbra a fészekferomont mutatják. Az öt forrás közül a legközelebbi behordását végzik a hangyák, és a fenti, valamint a jobb szélső felé is kialakulóban van a feromonnyom. Az ételferomon elhelyezkedéséből az is látszik, hogy minden forrásnál járt már legalább egy hangya.

7.4. A hangyakolónia rendezettsége

A vázolt algoritmus hatékonysága és flexibilitása nagyon vonzó, mivel a kollaboratív robotikai feladat célkitűzéseit teljesíti. Mégis, ahogy azt Simon is megjegyzi, a homokban járkáló hangya egyszerűen az egyenetlen felszín útvonalait követi, igen összetett mozgásmintázatot létrehozva ([157]). Ezért elképzelhető, hogy a gyűjtögetés hatékonyságában meglévő komplexitás valami módon a környezetbe ágyazódva létezik, a kezdeti rendezettségtől függ. (A rendezettség ezúttal az ételelhelyezkedésben meglévő szabályszerűséget, a nem-véletlenszerűséget jelenti.)

7.4.1. A hatékonyság függése a környezet rendezettségétől

A problémát számítógépes szimulációkon keresztül vizsgáltuk. Ezek során az F ételmennyiség különböző kezdőállapotokat mutatott. Az étel egyenletesen oszlott szét G darab ételforrás között, melyek véletlenszerűen helyezkedtek el L-en. Egy

Paraméter	Érték	Paraméter	Érték
N	100	w	0.1
S	100	α	1
R	5	δ	0.01
F	800	ρ	0.86
m	100	D	2

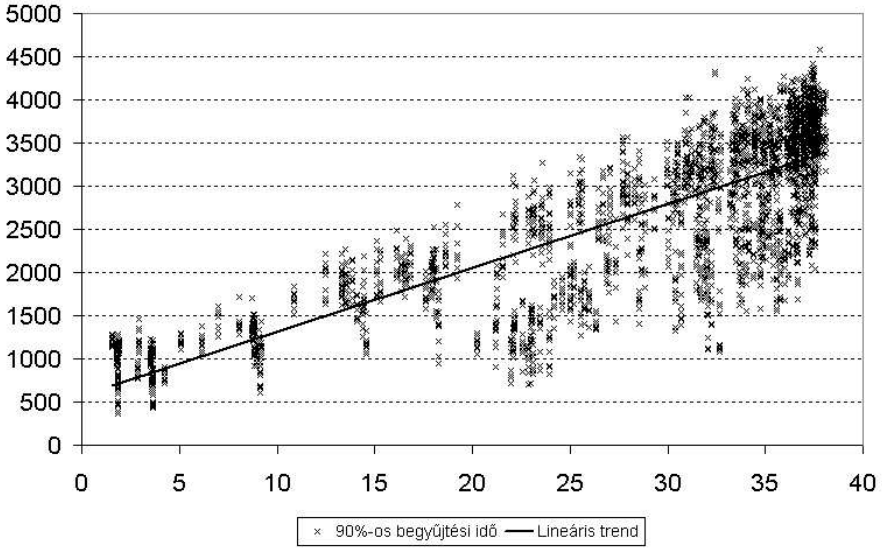
7.1. táblázat. A kísérletek paramétereinek értékei

ételforrás körül az eloszlás egy σ szórású kétdimenziós Gauss-eloszlásnak felelt meg. G értéke a tesztek során 1 és 10, σ-é 1 és 50 között változott. A modell egyéb paramétereit a 7.1. táblázat tartalmazza. Ezek az értékek minden futás esetén azonosak voltak, vagyis csupán a végrehajtandó feladat változott meg.

Az étel rendezettségének mérhetővé tételéhez az ételegységek közötti páronkénti távolságok átlagos értékét használtuk. Ez a mérték, mely nyilvánvalóan S és F függvénye is, növekszik a rendezettség csökkenésével. Ezzel összhangban a továbbiakban az ételkonfiguráció rendezetlenségének fogom hívni. (A tórusz méretétől és az étel mennyiségétől való függés a tesztekben figyelmen kívül hagyható, mivel a fenti két paraméter konstans.)

Ezen kívül a kolónia teljesítményének mértékeként az étel 90%-ának fészekbe hordásához szükséges szimulált időt veszem alapul. Ennek az az oka, hogy az utolsó néhány ételegység elhelyezkedése véletlenszerű az ételforrás környezetében, és ebből kifolyólag gyakran a véletlen bolyongás segítségével sikerül begyűjteni, mivel nem vezet hozzá feromonnyom. Az utolsó 10% begyűjtési idejét is figyelembe véve, a mérésekhez egy újabb „véletlen zaj” adódik. Ettől függetlenül az eredmények a teljes begyűjtés — mint teljesítménymérték — használatakor is érvényesek, még ha kevésbé kifejezett formában is.

A 7.4. ábra összegzi a hangyakolónia hatékonyságának függését a kezdeti ételkonfigurációtól. A vízszintes tengely a kezdeti rendezetlenséget mutatja, a függőleges tengely az étel 90%-ának begyűjtéséhez szükséges időt. Az nyilvánvaló, hogy a teljesítmény nagyjából lineárisan csökken (az étel begyűjtéséhez szükséges idő nő) a rendezetlenség növekedésével. Ugyanakkor a mérések szórása is megnő a rendezetlenséggel, ami némileg árnyalja a képet. Ebből kifolyólag a 7.5. ábrán a rendezetlenséget okozó két paraméter külön is vizsgálható: egyrészt az ételforrások számának emelése, másrészt a forrásokhoz tartozó szórás növelése. Az ábra az eredményeket $G > 1$ és $\sigma > 1$ értékekre külön mutatja. (Azok

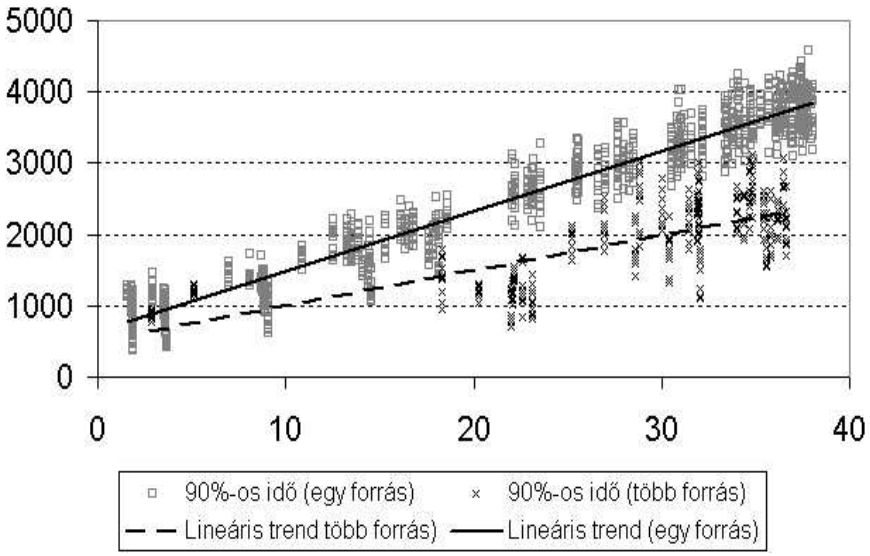


7.4. ábra. A hangyakolónia hatékonyságának függése az étel elhelyezkedésében tapasztalható kezdeti rendezettségtől. A vízszintes tengely a kezdeti rendezetlenség mértéke, a függőleges tengely a 90% begyűjtéséhez szükséges idő. Minden egyes kereszt a szimuláció korábban megadott paramétereivel készült egy-egy futást jelent, miközben az étel elhelyezkedését meghatározó paraméterek változtak: G 1, 2, 5 és 10 értékeket vett föl, míg σ 1, 2, 5, 10, 15 és 20 között változott. $G = 1$ esetén a tesztek σ 8, 20, 25, 30, 40 és 50 értékeire is elkészültek. Minden kombinációban az étel kezdeti elhelyezkedését 10 különböző pszeudovéletlenszám-generátor kezdőérték határozta meg, miközben a hangyák mozgását is 10 eltérő véletlen sorozattal vezéreltük. Vagyis egy G–σ kombinációhoz az ábrán 10×10 jel tartozik.

az esetek, amikor mindkét paraméter eltér 1-től, itt nem láthatók.)

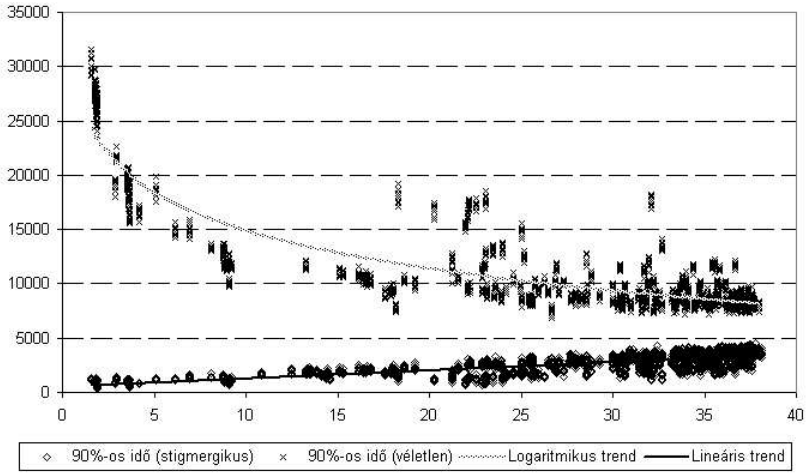
Látható, hogy a gyűjtögető hangyák a pontszerű ételforrásokat favorizálják, sőt, gyorsabban hordanak be a fészekbe több pontszerű ételforrást, mint egy elszórtat. Ennek az az oka, hogy a kevésbé elterülő forrás helye jobban kommunikálható (még ha indirekten is) a hangyapopuláción belül, szemben a szórt forrással. A pontosabban felismert források akár párhuzamosan is látogathatók, ami a jobb teljesítményt megmagyarázza.

Úgy tűnhet, hogy az eddigi eredmények nem mondanak többet, minthogy az algoritmus hatékonysága függ a feladat bonyolultságától. Bár ez triviálisan igaz, fontos látni, hogy ez a függés önmagában nem árulna el túl sokat arról, mely feladatok oldhatók meg hatékonyan a hangyaszerű osztott algoritmusokkal. Noha a



7.5. ábra. A hangyakolónia hatékonyságának függése az étel elhelyezkedésében tapasztalható kezdeti rendezettségtől, elkülönítve egy és több ételforrás esetét. A vízszintes tengely a kezdeti rendezetlenség mértéke, a függőleges tengely a 90% begyűjtéséhez szükséges idő. Minden egyes jel a szimuláció korábban megadott paramétereivel készült egy-egy futást jelent, miközben az étel elhelyezkedését meghatározó paraméterek változtak. Az üres négyzetek $G = 1$ értékhez tartoznak, miközben σ 1, 2, 5, 8, 10, 15, 20, 25, 30, 40 és 50 között változott. A keresztek esetén $\sigma = 1$ és G veszi föl sorra a 2, 5 és 10 értékeket. Minden kombinációban az étel kezdeti elhelyezkedését 10 különböző pszeudovéletlenszám-generátor kezdőérték határozta meg, miközben a hangyák mozgását is 10 eltérő véletlen sorozattal vezéreltük. Vagyis egy G–σ kombinációhoz az ábrán 10×10 jel tartozik.

legjobb, legrosszabb, átlagos teljesítmények kiértékelése gyakori az algoritmusok elemzésekor, a hangyaalgoritmusokra viszonylag nehezen alkalmazhatók ([135]). Vizsgálataink lényege inkább az, hogy a teljesítmény a kezdeti ételkonfiguráció rendezetlenségétől függ, és ez a kapcsolat nagyjából lineáris. A 7.6. ábra demonstrálja, hogy ez a fajta függés nem teljesen nyilvánvaló, összehasonlítva az eddigi eredményeket véletlen bolyongást végző, koordinálatlan hangyák viselkedésével. Az ábra fő üzenete nem az, hogy a stigmergikus hangyák jobban teljesítenek, mint a véletlenül bolyongók, hanem hogy a kétféle kolónia teljesítménye eltérő módon függ a kezdeti rendezetlenségtől.



7.6. ábra. A hangyaként viselkedő hangyakolónia és a véletlen bolyongást végző hangyák hatékonyságának összehasonlítása. A vízszintes tengely a kezdeti rendezetlenség mértéke, a függőleges tengely a 90% begyűjtéséhez szükséges idő. Az üres négyzetek a 7.4. ábrán látott futásokat jelentik, míg a keresztek azonos paraméterezés mellett végzett „véletlen” hangyák által elért eredmények. A véletlenül mozgó hangyákkal végzett tesztekhez a w érték 1-re nőtt.

7.4.2. A környezet és a kolónia rendezettségének összefüggései

A számítógépprogramok információkezelési problémákat oldanak meg. Más szóval a beérkező adatok konvertálását végzik, ami a bennük lévő információtartalom megváltozását jelenti. A gyűjtögető hangyák esetében is hasonlóról beszélhetünk: az ételt a környezetből a fészekbe szállítják, s a környezetben tárolt információtartalmat változtatják meg, ahogy azt a munkavégző robotok is teszik. Hasznos volna azt megállapítani, hogy a hangyakolónia feladata az étel rendezetlenségének csökkentése, de ez nem mindig teljesül. Különösen a legegyszerűbb esetben, amikor egy teljes ételforrást kell begyűjteni. Ekkor az étel rendezetlenségének változása az ételforrás és a fészek méretének egymáshoz való viszonyán múlik.

Ebből következően érdekes megvizsgálni, hogy a környezet rendezetlensége (más szempontból az információtartalma) miként változik a gyűjtögetés során, és hogy ez a változás miként tükröződik a kolónia rendezetlenségében. A továbbiakban vizsgálatainkat egyetlen ételforrás esetére korlátozzuk.

Az ételegység rendezetlenségét a korábbiakban bevezetett módon mérjük. Ehhez hasonlóan definiáljuk a hangyakolónia rendezetlenségét: ez a hangyák közötti

páronkénti távolságok átlagos értéke.

A 7.7. és a 7.8. ábra mutatja az étel és a hangyakolónia rendezetlenségének változását egy ételforrás esetére. Mindegyik kis képen folytonos vonallal látható az étel és szaggatott vonallal a hangyák rendezetlenségének változása 10 futásra, melyek során csak a hangyák mozgását irányító véletlen paraméter változott. Azonos sorokban az étel szórása a forrás körül megegyező (σ paraméter). Ez az érték föntről lefelé növekszik. A két ábrán az étel eltérő középpont körül szóródik.

A hangyakolónia kezdeti rendezetlensége mindenütt egyformán kicsi, mivel a hangyák a fészekből indulnak el. Ahogy elkezdik bejárni környezetüket, rendezetlenségük lényegesen megemelkedik. Az érték akkor lesz a legnagyobb, amikor valamelyikük megleli az ételforrást. Ezután a hangyák egy ösvényt építenek ki az ételforrás és a fészek között, ami rendezetlenségük csökkenését okozza. A maximális érték nagysága és időzítése az ételforrás helyétől függ: ha közel van a fészekhez, akkor a kezdeti véletlen bolyongás rövidebb ideig tart, a hangyák kevésbé szóródnak szét, így a csúcsosodás alacsonyabb, és előbb következik be. Az étel szórása is befolyásolja a hangyák legnagyobb rendezetlenségét. Ha a szórás nagyobb, akkor az étel jobban szétterül, így a forrás „pereme” közelebb lesz a fészekhez.

A gyűjtögetés végeztével, amikor az étel nagy része a fészekben van, a hangyakolónia rendezetlensége ismét megnő. Ez azért történik, mert már nincs elég étel, ami az összes hangyát foglalkoztatná, ezért ezek ismét felfedezésre indulnak. Azonban a feromonnyom továbbélése késlelteti a folyamatot, ahogy az a 7.7. és a 7.8. ábrák kis ételszórású első és második soránál megfigyelhető. A hangyák rendezetlensége ebben az utolsó szakaszban esetenként magasabb, mint a kezdeti szétszóródásnál. Ez két tényezőtől is függ. Először is attól, hogy a kolónia mennyi idő alatt találta meg először a forrást. Ha gyorsak voltak, akkor az első csúcs alacsony volt. Másrészről az is meghatározó, hogy a mérések mikor fejeződtek be, azaz a feladat elvégzése után még mennyi ideig fut a szimuláció. Amikor az összes étel a fészekbe jutott, akkor hosszú távon a hangyák rendezetlenségének el kell érnie a rendszer paraméterei által meghatározott elméleti maximumot. Ez abból következik, hogy a hangyák étel hiányában teljesen véletlenszerűen mozognak.

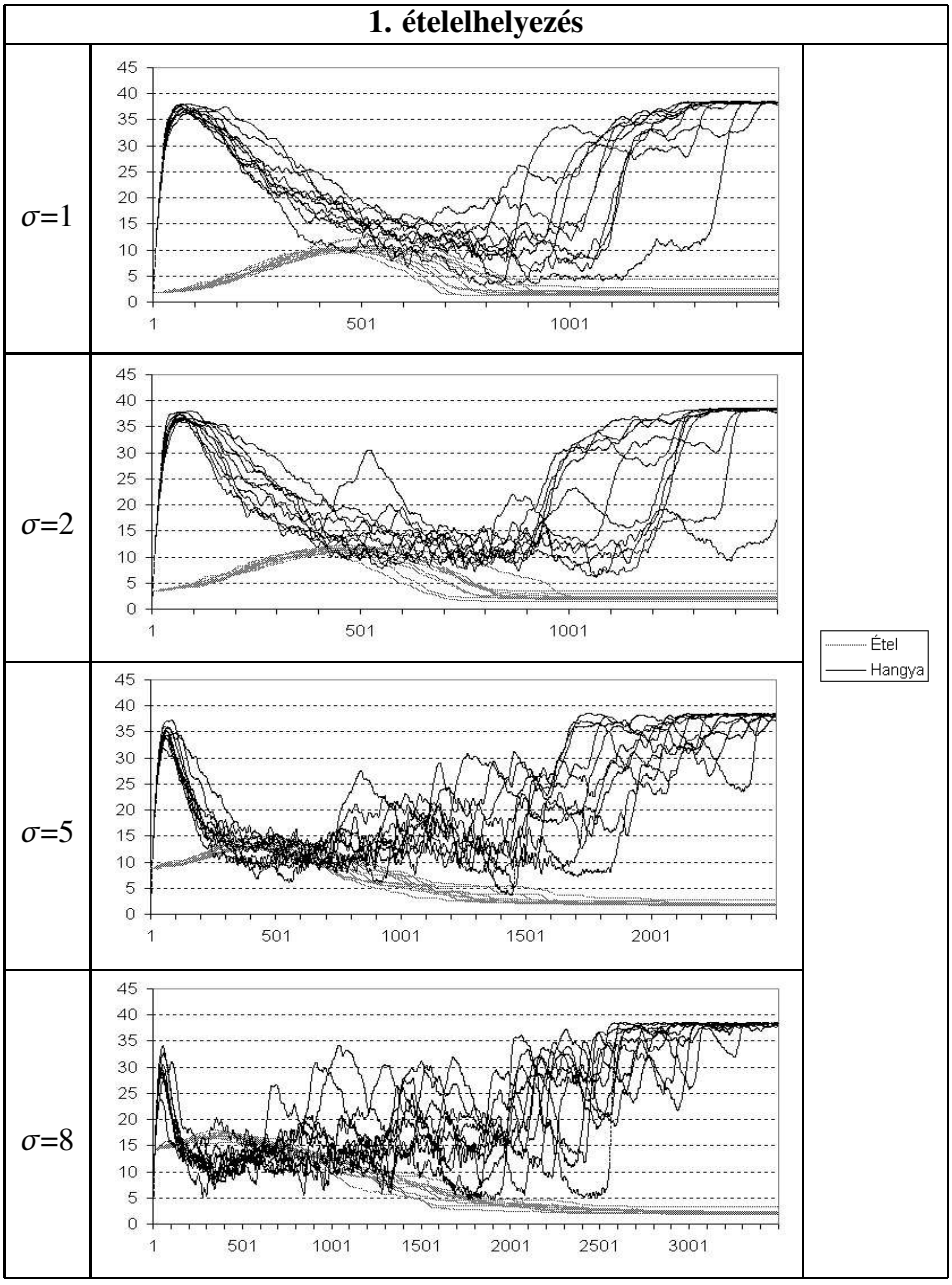
Az étel vizsgálatára áttérve megfigyelhető, hogy a kezdeti rendezetlenség a sorokon belül megegyezik, hiszen ez az egyetlen ételforrás körüli szórás para-

méterétől függ, ugyanakkor σ növelésével ez is növekszik. A kezdeti véletlen bolyongás időszaka után néhány hangya megtalálja az ételforrást, amitől az étel rendezetlensége elkezd megváltozni. Ahogy a hangyáké csökken, az ételé nőni kezd. Ez azért tapasztalható, mert az étel a forrás viszonylagosan rendezett kezdeti állapotától távolodik a fészek felé haladtában. A folyamat későbbi szakaszában az ételegységek három csoportba sorolhatók. Egy rész még mindig eredeti helyén, a forrás közelében van, némelyeket már behordtak a fészekbe, míg a többit éppen szállítják. Ez a tagozódás az oka a növekedő ételrendezetlenségnek. Miután azonban az étel fele már a fészekbe jutott, a további szállítások csökkentik a rendezetlenséget. Még azok az ételegységek is, amelyek úton vannak, mivel az étel nagyobb csoportja felé haladnak. (Nagyobb ételforrások esetén, vagyis amikor σ értéke magas, a kezdeti rendezetlenség annyira nagy, hogy az emelkedési fázis szinte teljesen eltűnik.)

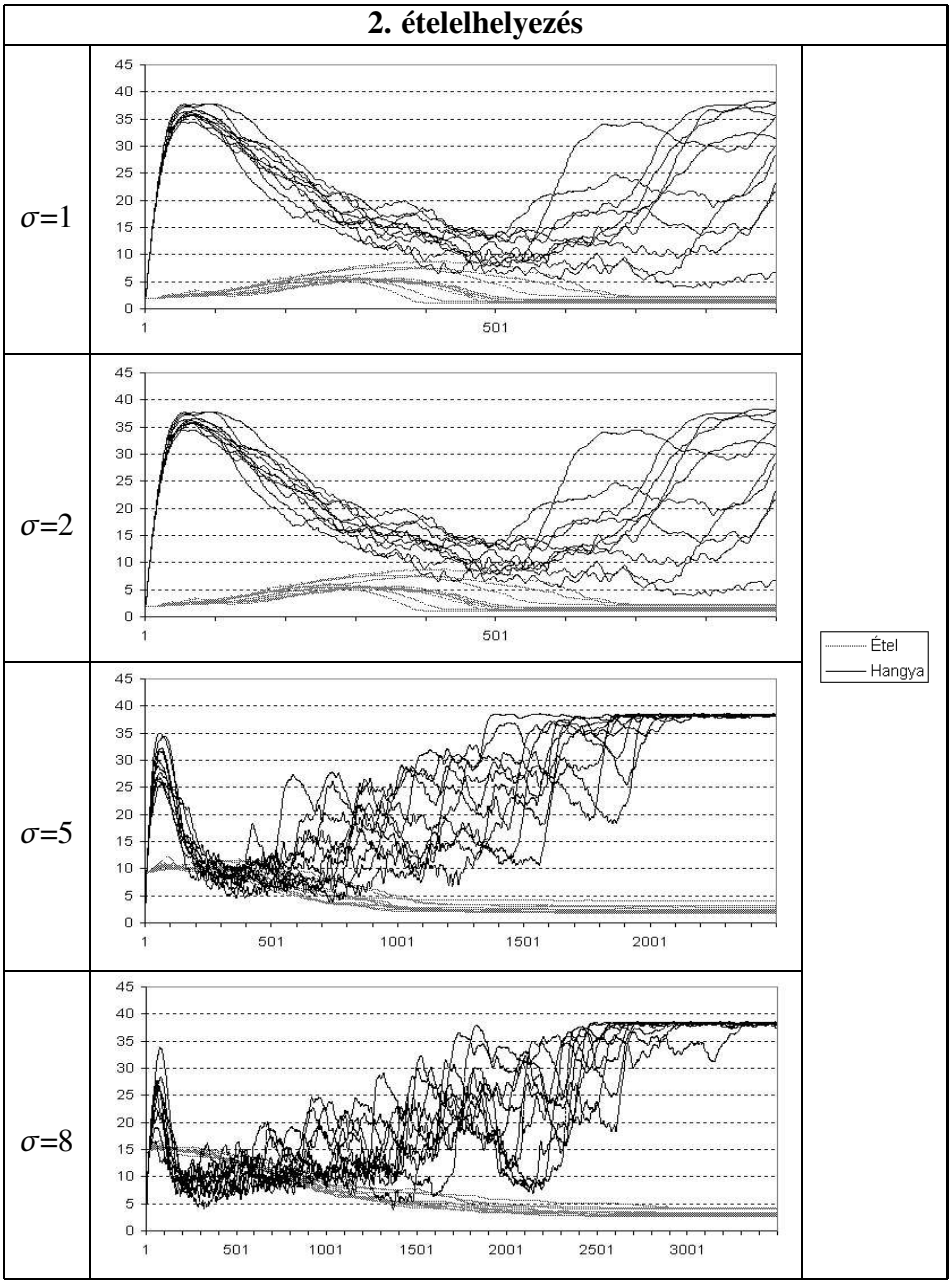
Egy lényeges megfigyelés, hogy a hangyakolónia nagyjából akkor éri el rendezettségének maximumát, amikor az étel rendezetlensége a maximumon van. Ez az az időszak, amikor a hangyák kialakították az optimális útvonalat az ételforrás és a fészek között, és oda-vissza járkálnak rajta, ételt szállítva. Ekkor a leginkább rendezett, vagyis koordinált a hangyakolónia viselkedése az egész eljárás során. Másrészről ekkor szállítják leginkább hatékonyan az ételt. Emiatt ekkor van a legtöbb ételegység mozgásban, az optimális út teljes hosszán. Ez magyarázatot ad az étel rendezetlenségének csúcsosodására is.

A hangyák minimális és az étel maximális rendezetlenségének tényleges értéke a fészek és a forrás távolságától függ. Az előbbi magyarázat alapján, ha feltételezzük, hogy a hangyák az optimális út mentén egyenletesen oszlanak el, akkor rendezetlenségük mértékét valójában az úthossz határozza meg. Az ételegységek rendezetlensége ebben a fázisban az ágensektől függ, hiszen a hangyák szállítják az étel nagy részét.

Amikor az ételforrás nagyobb, vagyis amikor σ értéke magasabb, a hangyakolónia rendezetlenségének növekedése korábban megkezdődik. Technikai értelemben ennek az az oka, hogy az útvonal leghatékonyabban két pontot köt össze: a forrás először megtalált részét a fészek ehhez legközelebb eső részével. Ha a forrás nagyméretű, akkor az eredeti végéhez közel az étel hamarabb elfogy, minthogy az egész forrás kimerülne, ezért a hangyák ismét felfedezésre kényszerülnek. Nyilván ezúttal könnyebb dolguk van, mint kezdetben, de azért néhány hangya másfelé kezd el bolyongani. Ez egyben magyarázat az eljárás végén a hangyák



7.7. ábra. Az étel (folytonos vonal) és a hangyakolónia (szaggatott vonal) rendezetlenségének változása egy ételforrás esetén. Mindegyik kép a mért értékek időbeli alakulását mutatja 10 futásra (különböző pszeudovéletlenszám-generátor kezdőértékkel a hangyákra vonatkozóan). A sorok az étel kezdeti rendezetlenségének növekvő szórású (σ) esetei.



7.8. ábra. Az étel és a hangyakolónia rendezetlenségének változása egy ételforrás esetén az étel egy újabb véletlen elhelyezésekor.

rendezetlenségének nagyobb változásaira és az utolsó 10% begyűjtésénél megfigyelt csökkenő hatásfokra is.

Amikor az összes étel a fészekbe van gyűjtve, az étel rendezetlensége felveszi végső értékét. Ez nagy σ esetén alacsonyabb, mint a kezdeti mennyiség. Ez azért van, mert ilyen esetekben az ételforrás kezdeti mérete nagyobb, mint a fészeké. Egyúttal a hangyák hajlamosak a fészek forrás felé eső részen letenni az ételt, a rendezetlenséget ezzel tovább csökkentve még kis szórások esetén is.

A 7.7. és a 7.8. ábra az ételforrás két véletlen elhelyezésű pozícióját mutatja. Az eredmények összehasonlításából jól látható, hogy a hangyakolónia a második esetben hatékonyabb. Ez a szórás aktuális értékétől függetlenül igaz, ugyanakkor egy adott konfigurációban a szórás növekedése növeli a futási időt, ahogy az az előbbiekben már látható volt.

Egy másik megfigyelés, hogy az étel növekvő szórása a futás egyre nagyobb fluktuációját okozza. Miközben az étel rendezetlenségének jellege nagyjából megegyezik, a hangyakolóniáé jobban változik a szórásnöveléssel. Ez különösen igaz a minimális rendezetlenség utáni szakaszra. Ez azért lényeges, mivel a hangyák valószínűségi algoritmust alkalmaznak. Mégis, a kis szórású, szinte pontszerű források esetén az algoritmus viselkedése szinte független az adott véletlen hangyaindítástól. Ugyanakkor nagy σ értékekre a hangyakolónia működése jobban múlik a véletlen tényezőkön. Észrevehető, hogy ez a függőség a gyűjtögető algoritmus sztochasztikus voltából fakad, nem pedig az étel elhelyezésében meglévő véletlen faktoron múlik, annak ellenére, hogy σ az utóbbit szabályozza közvetlenül.

Az eddigi elemzés egy ételforrás esetére korlátozódott. Több forrást hasonló módon lehetne vizsgálni. Amikor a forrásokat egymás után gyűjtik be a hangyák, akkor az étel rendezetlenségében több csúcsosodás is megfigyelhető. A futás vége felé jelentkező csúcsok gyakran hirtelen leesnek a végső, alacsony rendezetlenségű állapotba, nagyjából akkor, amikor a begyűjtött étel mennyisége elhagyja a $0.5F$-et. Megfigyelhető az is, hogy az ételrendezetlenség csúcsai nagyjából egybeesnek a hangyák rendezetlenségének völgyeivel, bár a forrás párhuzamos begyűjtésével a függés kevésbé hangsúlyos. Ezen kívül két forrás egymás utáni begyűjtése között a hangyák és az étel rendezetlensége egyaránt nő. Ez a hangyák — ideértve az éppen szállítókat is — felfedező viselkedéséből adódik, amikor egyik forrásról térnek át a másikra.

7.4.3. Kapcsolódó munkák

A társas rovarok viselkedésének tanulmányozása egyre több munkát inspirál az informatikában ([140]). A munkák tekintélyes része a hangyák gyűjtögető viselkedésének különféle módozataival foglalkozik. Dorigo és társai alapos áttekintést nyújtanak a téma elméletéről és gyakorlati alkalmazásáról, mely számos más feladat mellett magában foglalja az utazóügynök-problémát, a gráfszínezést, az útválasztást telekommunikációs hálózatokban és a taszkütemezést is ([50]). Egyúttal az eredetileg megfigyelt jelenséget is leírják, ahogy azt Deneubourg és társai vizsgálták a *Linepithema humile* fajtájú hangyáknál ([45]).

A hangyák gyűjtögető viselkedése alapján készített algoritmusokat a hangyakolónia-optimalizáció[3] nevű metaheurisztika gyűjti egybe és általánosítja ([37]). Bár ez a megközelítés túllép a Deneubourgék által vázolt modellen mind részletességben, mind alkalmazhatóságban, mi mégis az eredeti leírásnál maradtunk. Ennek az oka eltérő motivációnkban keresendő: mi a hangyák gyűjtögetésének működését szerettük volna értelmezni, amihez az egyszerűbb modell is elégségesnek bizonyult.

A fenti algoritmusok eddigi vizsgálata különféle konvergenciatulajdonságokat mutatott ki, így azt, hogy a hangyák közel optimális útvonalat építenek ki a fészek és az ételforrás között, illetve hogy másodrendű fázisátmenet köti össze a véletlen és a rendezett viselkedést ([29], [50]). Mégis viszonylag kevés kutatás foglalkozott azzal, hogy valójában miért működnek a hangyaalgoritmusok, pedig erre szükség volna, ha olyan „hangyaszerű”, stigmergikus algoritmusokat szeretnénk létrehozni, melyek nem közvetlen leképezései egy természetben megfigyelt jelenségnek.

Ramos és társai a környezet negatív és pozitív visszacsatolásainak hatását vizsgálták a hangyák lárvaválogató viselkedésére ([141]). Eddig a leglényegesebb próbálkozás Parunak és Brueckner nevéhez fűződik ([136]). Az ő megfigyelésük fontos mozzanata, hogy a hangyaszerű rendszerek több szinten értelmezendők: a globális, makroszintű önszerveződést a lokális, mikroszintű rendezetlenségnövekedés táplálja. (A témát formálisan is elemzi Bar-Yam, aki úgy találja, hogy a komplexitás összege a rendszer különböző szintjein egy bizonyos rögzített szabadságfokig változatlan, és független az adott rendszer változásától ([10])). Parunak és társai értelmezésében a makroszintű viselkedést a hangyák lépései je-

[3] Ant Colony Optimization (ACO)

lentik, míg a mikroszintet a feromonmolekulák mozgása jelképezi. Bár ehhez a megközelítéshez képest a saját vizsgálatok csupán a globális szint eseményeivel foglalkoznak, a kísérletek második csoportja is az ő magyarázatukat támasztja alá. Parunak és társai tanulmányozták azt az általánosabb problémát is, hogy miként lehet lokális döntések kontrolljával a globális célokat elérni ([137]). Állításuk szerint „ilyen rendszerek létrehozása jelenleg inkább művészet, mint tudomány". Cikkükben bemutatnak egy adaptív járási módot (a hangyaválogatási eljárás egy minimális változatát), és három gyakorlati alkalmazás lényeges tulajdonságait vezetik le a modell analíziséből. A mi munkánk is egy lépés a gyűjtögető algoritmus belső működésének megértése irányába.

Hasonló céllal vizsgálja Gutowitz a hangyák lárvaválogató algoritmusát ([68]). Szerinte a lokális döntések (a hangyák viselkedése) kellenek a globális hatékonysághoz. Ő alap- és komplexitáskereső hangyákat alkalmaz, megállapítva, hogy az utóbbiak megfelelőbbek a feladatra. Szemben Parunak és társai munkájával, Gutowitz szerint a rendezetlenséget a hangyák energiafogyasztása viszi be a kísérletbe, ami ellensúlyozza a hangyák munkája által növekvő környezeti rendezettséget. Gutowitz az energia hatékony felhasználására fókuszál, melyet a hangyák minden egyes lépése előtti döntéshez felhasznált idővel mér. Ez annyiban tér el munkánktól, hogy mi a környezet rendezettségét a kolónia rendezettségével vetjük össze.

A cikkben megfogalmazott gondolatok a kollektív robotikához is kapcsolhatók. Úttörő munkájukban Deneubourg és társai fölvetik, hogy a válogató és csoportosító hangyák mobil robotok viselkedési modelljéül szolgálhatnak ([46], [13]). Későbbi kísérletek a hangyák válogatásának teljesítményét vizsgálják. Martinoli és társai az egybehordott tárgyak átlagos csoportméretét elemzik az együttműködő robotok számának tükrében ([108]). Holland és társai a környezet méretével, a letevés valószínűségével és a szenzor áthangolásával kísérleteznek ([73]). A feladat általánosításával Handl és társai a stigmergikus csoportosító eljárást hasonlítják össze hagyományos klaszterező algoritmusokkal különféle mesterséges és természetes sokdimenziós adathalmazokon ([69]). Wilson és társai három gyűrűs válogatási algoritmust elemeznek a szétválasztás, a kompaktság, az alak és a teljesség tekintetében ([196]). Azonban ezen kísérletek egyike sem vizsgálja az algoritmus hatékonyságát a teljes bemenet jellegzetességeihez mérten. Ezen kívül Krieger és társai használnak robotokat az ételgyűjtő algoritmussal végzett kísérletekhez, de a teljesítményt a feladat függvényében ők sem vizsgálják ([89]).

7.4.4. A rendezettség és a rendezetlenség mérése

A kapcsolódó munkák közül több is foglalkozik az entrópia, az információszint, a rendezettség és a rendezetlenség kérdésével, ugyanakkor eltérő mértékeket használnak ezen mennyiségek mérésére. Parunak és társai a terminológiában meglévő kihívásokat is elemzik, vagyis az entrópia kifejezés kettőségét a termodinamika és az információelmélet területén ([136]). Bár a Shannon által definiált utóbbi formálisan nagyon hasonlít az előbbihez, valódi kapcsolatuk nem teljesen tisztázott ([153]).

Parunak és társai az információ rendezetlenségének egy térbeli változata mellett teszik le voksukat, és vizsgálják, hogy a mesterségesen megválasztott, a környezetre illesztett háló mérete mennyiben befolyásolja az eredményeket. Gutowitz ezzel szemben két eltérő mértéket használ a mikro- és a makroszint rendezetlenségének mérésére. Mikroszinten a bonyolultság a tárgyak sűrűségét jelenti egy adott pontban és környezetében, makroszinten pedig ő is térbeli rendezetlenséget használ, és az eredmények felbontástól való függéséről ír.

A kísérletek elvégzése előtt mi is számtalan bonyolultsági mértéket vettünk számba mind a hangyakolónia, mind az étel rendezetlenségének elemzéséhez. Az elméleti alaposság jegyében Shannon eredeti definíciója jó jelöltnek tűnt. Ökológusok és demográfiai kutatók gyakran használják ezt a mértéket egy terület homogenitásának vizsgálatára, hogy bizonyos fajok diverzitását megállapítsák, vagy a területi tagozódást mérjék ([144], [180]). Azonban ez a mérték, ahogy arra Gutowitz és Parunak is rámutat, a felbontáson keresztül egy erős, mesterséges függést vezet be, melyet mi szerettünk volna elkerülni. Ráadásul olyan mértéket kerestünk, amivel az ételgyűjtés problémája is jól kifejezhető.

Egy másik gyakran használt megoldás a rendezetlenség mérésére a rendszer trajektóriájának összehasonlítása egy előre meghatározott alapműködéssel. Ez a megközelítés az arcfelismerés közben megfigyelhető szaggatott szemmozgás kezelésére alkalmasnak bizonyult ([95]). Esetünkben ez az étel és a hangyakolónia rögzítését jelenti egy konfigurációban, majd az aktuális állapot összehasonlítását a rögzítettel. Azonban mivel a hangyakolónia és az étel állapota párhuzamosan módosul, változásuk az alapesethez képest kevésbé jellemző, ezért ezt a módszert egyelőre elvetettük.

7.4.5. Következtetések és további feladatok

A társas rovarok által inspirált algoritmusok az elmúlt évtizedben váltak népszerűvé. Az elterjedés abból a növekvő igényből fakad, hogy emergens jelenségek felhasználásával lehessen megoldani a ma és a holnap szoftvermérnöki problémáit. Egyre több alkalmazás használja a hangyakolóniákat, miközben egy általánosított, jól formalizált metaheurisztika, a hangyakolónia-optimalizáció is körvonalazódott. Mégis, ezen algoritmusok többsége lényegében ismert és részletesen bemutatott rovarmodelleket használ, úgymint gyűjtögetést, válogatást vagy feladatmegosztást. A hangyakolónia-optimalizáció általánosítja ugyan a gyűjtögetés modelljét, így alkalmassá téve egy szélesebb problémakör kezelésére, de nem mond semmit arról, hogy miért és milyen körülmények között működik ez a heurisztika. Ahogy azt korábban már említettük, viszonylag kevés cikk foglalkozik a problémával, és a kérdés alapvetően megválaszolatlan.

Ebben a fejezetben egy rendezetlenségi mértéket használtunk a modell elemzéséhez. Első lépésként azt vizsgáltuk, hogy a hangyakolónia teljesítménye miként függ az étel környezetbeli elhelyezkedésétől, ami a feladat bonyolultságának egyfajta mértéke. Azt találtuk, hogy a végrehajtási idő nagyjából lineárisan függ a kezdeti rendezetlenségtől. Ezen kívül azt tanulmányoztuk, hogy hogyan változik az étel rendezetlensége és a hangyakolónia koordinációja az ételgyűjtés során egy ételforrás esetén. Fő eredményünk annak kimutatása, hogy a hangyakolónia legnagyobb rendezettségét nagyjából akkor éri el, amikor az étel rendezetlensége a legnagyobb. Ez az az időszak, amikor optimalizált ösvény alakul ki az ételforrás és a hangyafészek között. Ezen kívül az egyedi ételforrás kezdeti rendezetlenségének növekedése hatására a kolónia teljesítménye jobban függ az egyes hangyák viselkedésében meglévő véletlen tényezőtől.

Az eredmények azt mutatják, hogy az ilyen típusú elemzésnek szerepe lehet a stigmergikus algoritmusok hajtóerejének megértésében. Ezután pedig új típusú, rovarszerű osztott algoritmusokat lehetne tervezni összetett problémák megoldásához. Bár hosszú távú célunk ez, előtte még további kísérletek elvégzését tervezzük. Először is szeretnénk az elemzést kiterjeszteni a két feromonmező vizsgálatára. Mivel ezek a mezők adnak információt az étel elhelyezkedéséről a hangyák számára, ezért ezek a mérések rávilágíthatnak arra, hogy miért olyan hatékony az információáramlás a stigmergikus rendszerekben. Ezen kívül egyéb rendezetlenségmértékek használatával, köztük a klasszikus térbeli entrópiáéval kideríthető,

hogy eredményeink mennyire függenek az aktuális mérték megválasztásától. Végül, egy későbbi fázisban tervezzük módszerünk kiterjesztését egyéb stigmergikus algoritmusokra, például válogatásra.

Összegzés

Értekezésemben intelligens ágensek navigációját vizsgáltam többféle nézőpontból. A feladat sikeres megoldása az állatok esetében a túléléshez, mobil robotok esetében a hatékony munkavégzéshez elengedhetetlen, ezért jelenleg is aktívan kutatott tudományterület. A navigációt végző robotnak a nehézségek autonóm, robusztus megoldását akár tetszőleges, ismeretlen környezetben is valós időben kell elvégeznie, lehetőleg univerzális módon. Jelenleg ez a mobil robotika egyik legfontosabb, egyelőre csak részlegesen megoldott problémája. A számítógépes szimulációk segítségével a tájékozódással kapcsolatos kísérletek virtuális térben folyhatnak. Ez, azon kívül, hogy költség-, idő- és energiatakarékosabb, jóval rugalmasabb megoldás, melyet utóbb a valódi, természetes környezetben végzett kísérletek igazolhatnak.

A szimulált ágensek navigációjával kapcsolatos vizsgálataim három csoportra oszthatók.

A robotszimulációs verseny során környezetreprezentáció nélküli intelligens viselkedés létrehozásával foglalkoztam. Készítettem egy valós időben működő, reaktív, moduláris felépítésű robotirányító programot a Webots szimulátorban, mely az 1999-es Artificial Life Creators Contest nemzetközi versenyen második, a 2000-esen pedig első helyet ért el.

A Webots szimulátorban végzett további kísérletek a különféle térképkészítési és útvonaltervezési eljárások működésének elemzéséről, összehasonlításáról szólnak. Létrehoztam egy foglaltsági hálón alapuló térképkészítő és értékiterációt használó útvonaltervező eljárást a Webots szimulátorban, mely ultrahangos érzékelői segítségével kreálja meg a modellezett környezetek térképét, és sikeresen járja be a különféle terepeket. Kiegészítettem a foglaltsági hálón alapuló térképezést egy topológiai gráfot létrehozó eljárással, mely az értékiterációs útvonaltervezést váltja fel. Megmutattam, hogy az új módszer hatékonyabb terepbejárást tesz lehetővé. A foglaltsági hálóként reprezentált térkép elkészítéséhez az

ultrahangos érzékelő mellé bevezettem a kamera használatát, mely kinézetalapú akadályérzékelést biztosít, és a topológiai gráf alapú útvonaltervezéssel általában jobban teljesít, mint a korábbi eljárások.

A hangyák ételgyűjtésével foglalkozó harmadik rész a feladatvégzés hatékonyságának és teljes folyamatának környezetfüggését vizsgálja. Hangyák stigmergikus ételgyűjtő viselkedésének Deneubourg és társai által készített modelljében vizsgáltam az ételforrások megtalálásának és begyűjtésének hatékonyságát a környezet konfigurációjának függvényében. Beláttam, hogy a kezdeti rendezetlenségtől nagyjából lineárisan függ a hangyaraj teljesítménye. Kimutattam, hogy az ételgyűjtés során a hangyák legnagyobb rendezettségének állapota nagyjából akkor következik be, amikor az étel a leginkább rendezetlen. Továbbá megfigyeltem, hogy a környezet kezdeti konfigurációjának összetettebbé válása — vagyis az étel szórásának növekedése — fölerősíti a hangyák viselkedésében meglévő véletlen szerepét.

Summary

My Ph.D. research focused on the navigation of simulated intelligent agents. Successful solution of the navigation task is inevitable for the survival of animals, and for the efficient work of mobile robots. This fact explains why the discipline is an active research domain. A navigating robot has to provide a universal, autonomous, robust, and real-time solution of the problem. This is one of the most important open questions of mobile robotics. Agent-based simulation is a unique tool for performing navigation-related experiments in a virtual world. Modeling robots saves development and execution time, cost, and energy. Further tests in real-world, natural environments could validate this approach.

This thesis presents contributions to the navigation of simulated agents in three main areas.

I developed a real-time, reactive, modular robot without explicit environment representation working in the Webots simulator. The controller was the runner-up of the 1999 and the winner of the 2000 Artificial Life Creators Contest international robot competition.

Further researches performed in the Webots simulator deal with the functioning and comparison of various map building and path planning methods. I implemented an occupancy grid based map building algorithm using value iteration as path planning. The robot employs ultrasonic range sensors to produce the map of the modeled environment and then successfully explores various test terrains. Later I appended a topological graph based path planning algorithm to the occupancy grid map replacing value iteration. I demonstrated that the new solution enables more efficient environment exploration. I extended the occupancy grid map building with the utilization of camera images. This approach integrates former obstacle avoidance using ultrasonic range finder with an appearance-based method. Map building with both sensor types in the context of topological graph path planning generally results faster exploration.

The third part of the dissertation is dedicated to the investigation of the performance and the development of food collecting behavior of social agents. I explored the relationship between the initial disorder in the environment and the performance of the ant foraging behavior in the model of Deneubourg et al. and I found that the performance of the anthill depends about linearly on the initial disorder of food. I further studied how the configuration of the task governs the level of coordination in the behavior of the ant colony and I demonstrated that during the development of the process the minimal disorder of the ants occurs about the same time when the food disorder peaks. Finally, I presented that growing initial distribution of the food amplifies the stochastic nature of the algorithm expressed in the random behavior of the ants.

Irodalomjegyzék

[1] A. Aboshosha. Adaptation of rescue robot behaviour in unknown terrains based on stochastic and fuzzy logic approaches. *Proceedings of IEEE / RSJ International Conference on Intelligent Robots and Systems IROS 2003*, 2003.

[2] D. J. Ahlgren and I. M. Verner. Conceptualising educational approaches in introductory robotics. *International Journal of Electrical Engineering Education*, 2004.

[3] M. A. Arbib, P. Érdi, and J. Szentágothai. *Neural Organization: Structure, Function, and Dynamics*. The MIT Press, Cambridge, MA, 1998.

[4] L. R. Aronson. Further studies on orientation and jumping behavior in the gobiid fish, bathygobius soporator. *Ann NY Acad Sci*, 188:378–392, 1971.

[5] R. Axelrod. The evolution of cooperation. *Science*, 211(4489):1390–96, 1981.

[6] R. Axelrod. *The Evolution of Cooperation*. Basic Books, 1984.

[7] T. Balch. *Behavioral Diversity in Learning Robot Teams*. PhD thesis, College of Computing, Georgia Institute of Technology, 1998.

[8] T. Balch and C. Arkin. Communication in reactive multiagent robotic systems. *Autonomous Robots*, 1(1):1–25, 1994.

[9] R. Balogh. I am a robot - competitor: A survey of robotic competitions. *International Journal of Advanced Robotic Systems*, 2(2), 2005.

[10] Y. Bar-Yam. Multiscale complexity/disorder. *Advances in Complex Systems*, 7(1):47–63, 2004.

[11] P. H. Batavia and I. Nourbakhsh. Path planning for the Cye personal robot. In *Proceedings of International Conference on Intelligent Robots and Systems*, volume 1, pages 15–20, 2000.

[12] J. A. Batlle, J. M. Font, and J. Escoda. Dynamic positioning of mobile robot using a laser-based goniometer. In *5th IFAC/EURON Symposium on Intelligent Autonomous Vehicles*, 2004.

[13] R. Beckers, O.E. Holland, and J.-L. Deneubourg. From local actions to global tasks: Stigmergy and collective robotics. In R. Brooks and P. Maes, editors, *Proceedings of the Fourth Workshop on Artificial Life*, pages 181–189. MIT Press, 1994.

[14] R. D. Beer. *Intelligence as Adaptive Behavior: an Experiment in Computational Neuroethology*. Academic Press, 1990.

[15] M. Bohus, D. Muszka, and P. G. Szabó. A szegedi informatikai gyűjtemény – in memoriam Kalmár László. *KöMaL Ifjúsági Ankét*, 2005.

[16] J. Borenstein, H. R. Everett, L. Feng, and D. Wehe. Mobile robot positioning: Sensors and techniques. *Journal of Robotic Systems*, 14(4):231–249, 1997.

[17] J. Borenstein and L. Feng. Correction of systematic odometry errors in mobile robots. In *1995 IEEE International Conference on Robotics and Automation*, 1995.

[18] J. Borenstein and Y. Koren. The vector field histogram - fast obstacle avoidance for mobile robots. *IEEE Transactions on Robotics and Automation*, 7(3):278–288, 1991.

[19] G. Borgefors. Distance transformations in digital images. *Computer Vision, Graphics, and Image Processing*, 34:344–371, 1986.

[20] V. Braitenberg. *Vehicles: Experiments in Synthetic Psychology*. MIT Press, 1984.

[21] J. E. Bresenham. Algorithm for computer control of a digital plotter. *IBM Systems Journal*, 1965.

[22] R. A. Brooks. A robot that walks; emergent behaviours from a carefully evolved network. *MIT, AI MEMO 1091*, 1989.

[23] R. A. Brooks. Intelligence without reason. In J. Myopoulos and R. Reiter, editors, *Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI-91)*, pages 569–595, Sydney, Australia, 1991. Morgan Kaufmann publishers Inc.: San Mateo, CA, USA.

[24] R. A. Brooks. Intelligence without representation. *Artificial Intelligence*, 47:139–159, 1991.

[25] R. A. Brooks. Artificial life and real robots. In *European Conference on Artificial Life*, pages 3–10, 1992.

[26] R. A. Brooks. *Flesh and Machines : How Robots Will Change Us*. Vintage Books, New York, paperback edition, 2003.

[27] W. Burgard, A. Derr, D. Fox, and A. B. Cremers. Integrating global position estimation and position tracking for mobilerobots: the Dynamic Markov Localization approach. In *Proc. of the IEEE/RSJ InternationalConference on Intelligent Robots and Systems*, 1998.

[28] Z. Byers, M. Dixon, K. Goodier, C. M. Grimm, and W. D. Smart. An autonomous robot photographer. In *Proceedings of the International Conference on Robots and Systems (IROS 2003)*, volume 3, pages 2636–2641, Las Vegas, NV, October 2003.

[29] D. R. Chialvo and M. M. Millonas. How swarms build cognitive maps. In L. Steel, editor, *The Biology and Technology of Intelligent Autonomous Agents*, volume 144, pages 439–450. Nato ASI Series, 1995.

[30] K. Chokshi, C. Panchev, S. Wermter, and K. Burn. Self organising neural place codes for vision based robot navigation. In *Proceedings of the International Joint Conference on Neural Networks*, pages 2501–2506, 2004.

[31] H. Choset. *Sensor Based Motion Planning: The Hierarchical Generalized Voronoi Graph*. PhD thesis, Carnegie Mellon University, 1996.

[32] T. S. Collett. Insect navigation en route to the goal: Multiple strategies for the use of landmarks. *The Journal of Experimental Biology*, 199:227–235, 1996.

[33] N. Collier, T. Howe, and M. North. Onward and upward: The transition to Repast 2.0. In *Proceedings of the First Annual North American Association for Computational Social and Organizational Science Conference*, 2003.

[34] J. Connell. *A colony architecture for an artificial creature*. PhD thesis, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1989. Technical Report AI-TR 1151.

[35] R. Conte. Multi-agent based social simulation. In *Agents Everywhere – Proceedings of the First Hungarian National Conference on Agent Based Computing*, pages 164–180, 1999.

[36] R. Conte and C. Castelfranchi. *Cognitive and social action*. University College of London Press, 1995.

[37] O. Cordón, F. Herrera, and T. Stützle. A review on the ant colony optimization metaheuristic: Basis, models and new trends. *Mathware & Soft Computing*, 9, 2002.

[38] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Algoritmusok*. Műszaki Könyvkiadó, Budapest, 1999.

[39] N. Correll. Collaborative exploration and coverage. Master's thesis, Collective Robotics Group, California Institute of Technology/Institute for Automatic Control, Swiss Federal Institute of Technology, Zurich, 2003.

[40] D. Csetverikov et al. Basic algorithms for digital image analysis: a course. Technical report, Eötvös Loránd University, Institute of Informatics, 2001.

[41] V. Csányi. *Etológia*. Nemzeti Tankönyvkiadó Rt., 1994.

[42] Cyberbotics S.a r.l. *Webots 2.0 User Guide*, 1999.

[43] P. Davidsson. Agent Based Social Simulation: A Computer Science View. *Journal of Artificial Societies and Social Simulation*, 5(1), 2002.

[44] P. A. DeBitetto, E. N. Johnson, M. C. Bosse, and C. A. Trott. The Draper Laboratory small autonomous aerial vehicle. In *Proceedings of the Enhanced and Synthetic Vision Conference, The International Society for Optical Engineering Conference*, volume 3088, pages 110–120, 1997.

[45] J.-L. Deneubourg, S. Aron, S.Goss, and J.-M. Pasteels. The self-organizing exploratory pattern of the argentine ant. *J. Insect Behavior*, 3:159–168, 1990.

[46] J-L. Deneubourg, S. Goss, N. Franks, A. Sendova-Franks, C. Detrain, and L. Chretien. The dynamics of collective sorting: Robot-like ant and ant-like robot. In J. A. Mayer and S.W. Wilson, editors, *Simulation of Adaptive Behavior: From Animals to Animats*, pages 356–365. MIT Press, 1991.

[47] D. Dennett. True believers. In *The Intentional Stance*. MIT Press, Cambridge, Mass., 1987.

[48] A. Diosi, G. Taylor, and L. Kleeman. Interactive SLAM using laser and advanced sonar. In *Proceedings of 2005 IEEE ICRA*, pages 1103–1108, 2005.

[49] G. Dissanayake, H. F. Durrant-Whyte, and T. Bailey. A computationally efficient solution to the simultaneous localisation and map building (SLAM) problem. In *ICRA*, pages 1009–1014, 2000.

[50] M. Dorigo, G. D. Caro, and L. M. Gambardella. Ant algorithms for discrete optimization. In *Proceedings of Artificial Life 5*, pages 137–172, 1999.

[51] A. Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46–57, 1989.

[52] P. Érdi and M. Lengyel. Matematikai modellek az idegrendszer-kutatásban. In Cs. Pléh, Gy. Kovács, and B. Gulyás, editors, *Kognitív idegtudomány*, pages 126–148. Osiris: Budapest, 2003.

[53] H. Everett. *Sensors For Mobile Robots Theory and Application*. A K Peters Ltd., 1995.

[54] I. A. Ferguson. *TouringMachines: An Architecture for Dynamic, Rational, Mobile Agents*. PhD thesis, Clare Hall, University of Cambridge, 1992.

[55] R. Fikes and N. Nilsson. STRIPS: a new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208, 1971.

[56] D. Fox, W. Burgard, and S. Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics and Automation*, 4(1), 1997.

[57] M. Franz and H. Mallot. Biomimetic robot navigation. *Robotics and Autonomous Systems*, 30:133–153, 2000.

[58] M. O. Franz, B. Schölkopf, P. Georg, H. A. Mallot, and H. H. Bülthoff. Learning view graphs for robot navigation. In W. Lewis Johnson and Barbara Hayes-Roth, editors, *Proceedings of the First International Conference on Autonomous Agents (Agents'97)*, pages 138–147, New York, 5–8, 1997. ACM Press.

[59] I. Futó, editor. *Mesterséges intelligencia*. Aula Kiadó, 1999.

[60] A. J. Garvey and V. Lesser. Design-to-time real-time scheduling. *IEEE Transactions on Systems, Man and Cybernetics*, 23(6):1491–1502, November/December 1993.

[61] B. Gerkey, R. T. Vaughan, and A. Howard. The Player/Stage project: Tools for multi-robot and distributed sensor systems. In *Proceedings of the 11th International Conference on Advanced Robotics (ICAR'03)*, 2003.

[62] D. Goldberg and M. J. Mataric. Interference as a tool for designing and evaluating multi-robot controllers. In *AAAI/IAAI*, pages 637–642, 1997.

[63] P. P. Grassé. La reconstruction du nid et les coordinations interindividuelles chez bellicositermes natalensis et cubitermes sp., la théorie de la stigmergie: essai d'interprétation du comportement des termites constructeurs. *Insectes Sociaux*, 6:41–81, 1999.

[64] A. Guillot and J.-A. Meyer. Synthetic animals in synthetic worlds. In Kunii et Luciani, editor, *Synthetic Worlds*. Wiley and sons, 1996.

[65] L. Gulyás. *Understanding Emergent Social Phenomena: Methods, Tools, and Applications for Agent-Based Modelling*. PhD thesis, Ph.D. School of Informatics, Eötvös Loránd University, 2004.

[66] L. Gulyás, L. Laufer, and R. Szabó. An information theoretic approach to stigmergy: The case of foraging ants. In *Proceedings of AISB'06: Adaptation in Artificial and Biological Systems*, volume 3, page 203, 2006.

[67] L. Gulyás, L. Laufer, and R. Szabó. Measuring stimergy: The case of foraging ants. In *Proceedings of the Fourth International Workshop on Engineering Self-Organizing Applications (ESOA 2006/AAMAS 2006, May 2006, Hakodate, Japan*, pages 76–91, 2006.

[68] H. Gutowitz. Complexity-seeking ants. In Deneubourg and Goss, editors, *Proceedings of the Third European Conference on Artificial Life*, 1993.

[69] J. Handl, J. Knowles, and M. Dorigo. On the performance of ant-based clustering. In *Proceedings of the Third International Conference on Hybrid Intelligent Systems*. IOS Press, 2003.

[70] J. Henderson, T. A. Hurly, M. Bateson, and S. D. Healy. Timing in free-living rufous hummingbirds, selasphorus rufus. *Current Biology*, 16(5):512–515, 2006.

[71] M. Hewett. The impact of perception on agent architectures. In *Proceedings of the AAAI-98 Workshop on Software Tools for Developing Agents*, 1998.

[72] J. Hoffmann, M. Jüngel, and M. Lötzsch. A vision based system for goal-directed obstacle avoidance. In *RoboCup 2004*, pages 418–425, 2004.

[73] O. Holland and C. Melhuish. Stigmergy, self-organisation, and sorting in collective robotics. *Artificial Life*, 5:173–202, 2000.

[74] A. Hopp, D. Schulz, W. Burgard, D. Fellner, and A. Cremers. Virtual reality visualization of distributed teleexperiments. In *Proceedings of the 24th Annual Conference of the IEEE Industrial Electronics Society (IECON'98)*, 1998.

[75] I. Horswill. Visual collision avoidance by segmentation. In *ARPA94*, pages II:1135–1141, 1994.

[76] Z. Istenes. Robotika: átmenet az oktatásból a kutatás felé. Technical report, ELTE, Informatika Kar, 2007. VII. Országos Neveléstudományi Konferencia.

[77] M. Iványi, L. Gulyás, and R. Szabó. Agent-based simulation in disaster management. In *Proceedings of the First International Workshop on Agent Technology for Disaster Management (ATDM 2006/AAMAS 2006), May 2006, Hakodate, Japan*, pages 153–154, 2006.

[78] A. K. Jain, editor. *Fundamentals of Image Processing*. Prentice-Hall,NJ, 1989.

[79] N. Jakobi. *Minimal Simulations for Evolutionary Robotics*. PhD thesis, School of Cognitive and Computing Sciences, University of Sussex, 1998.

[80] N. R. Jennings. An agent-based approach for building complex software systems. *Communications of the ACM*, 44(4):35–41, 2001.

[81] I. Kamon and E. Rivlin. Sensory-based motion planning with global proofs. *IEEE Trans. Robot. & Autom.*, 13(6):814–822, 1997.

[82] Gy. Kampis. The natural history of agents. In *Agents Everywhere – Proceedings of the First Hungarian National Conference on Agent Based Computing*, pages 10–24, 1999.

[83] Gy. Kampis. Az elme dinamikus modelljei. In J. Gervain and Cs. Pléh, editors, *A megismerés vizsgálata*. Osiris-Láthatatlan Kollégium, 2000.

[84] M. Khatib and R. Chatila. An extended potential field approach for mobile robot sensor-based motions. In *Proceedings of the Intelligent Autonomous Systems IAS-4*, pages 490–496, 1995.

[85] H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, and E. Osawa. RoboCup: The robot world cup initiative. In W. L. Johnson and B. Hayes-Roth, editors, *Proceedings of the First International Conference on Autonomous Agents (Agents'97)*, pages 340–347, New York, 5–8, 1997. ACM Press.

[86] J. Kodjabachian and J-A. Meyer. Evolution and development of neural networks controlling locomotion, gradient-following, and obstacle-avoidance

in artifical insects. *IEEE Transactions on Neural Networks*, 9:796–812, 1998.

[87] M. Koes, I. Nourbakhsh, and K. Sycara. Communication efficiency in multi-agent systems. *Proceedings of ICRA 2004*, 2004.

[88] D. Kortenkamp, R. P. Bonasso, and R. Murphy. *Artificial Intelligence and Mobile Robots*. AAAI Press/The MIT Press, 1998.

[89] M. Krieger, J.-B. Billeter, and L. Keller. Ant-like task allocation and recruitment in cooperative robots. *Nature*, 406(6799):992–995, 2000.

[90] B. Kröse and M. Eecen. A self-organizing representation of sensor space for mobile robot navigation. In *Proceedings of the IEEE/RSJ/GI Int. Conf. on Intelligent Robots and Systems IROS' 94*, pages 9–14, 1994.

[91] E. Kubinyi, Á. Miklósi, F. Kaplan, and V. Csányi. Hogyan kapcsolódhat össze a kutyakutatás és a társrobot-fejlesztés? In Gy. Kampis and L. Ropolyi, editors, *Evolúció és megismerés*, pages 329–342. Typotex Kft., 2001.

[92] B. Kuipers and Y-T. Byun. A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. *Journal of Robotics and Autonomous Systems*, 8:47–63, 1991.

[93] J-C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Norwell, MA, 1991.

[94] A. Lazanas and J.C. Latombe. Landmark-based robot navigation. *Algorithmica*, 13:472–501, 1995.

[95] T. S. Lee and S. X. Yu. An information-theoretic framework for understanding saccadic eye movements. *Neural Information Processing Systems*, 1999.

[96] W-P. Lee, J. Hallam, and H. Lund. Applying genetic programming to evolve behavior primitives and arbitrators for mobile robots. In *Proceedings of IEEE 4th International Conference on Evolutionary Computation*, volume 1. IEEE Press, 1997.

[97] J. Leonard and H. Feder. A computationally efficient method for large-scale concurrent mapping and localization. In D. Koditschek J. Hollerbach, editor, *International Symposium on Robotics Research*, 1999.

[98] J. J. Leonard and H. F. Durrant-Whyte. *Directed sonar sensing for mobile robot navigation*. Kluwer Academic Publishers, 1992.

[99] J. J. Leonard, P. M. Newman, R. J. Rikoski, J. Neira, and J. D. Tardos. Towards robust data association and feature modeling for concurrent mapping and localization. In *Proceedings of the 10th International Symposium on Robotics Research*, 2001.

[100] P. U. Lima and L. M. M. Custodio. Artificial intelligence and systems theory: Applied to cooperative robots. *International Journal of Advanced Robotic Systems*, 1(3):141–148, 2004.

[101] L. Lorigo, R. A. Brooks, and W. E. L. Grimson. Visually guided obstacle avoidance in unstructured environments. Technical report, IEEE Conference on Intelligent Robots and Systems, September 1997.

[102] F. Lu and E. Milios. Robot pose estimation in unknown environments by matching 2D range scans. In *CVPR94*, pages 935–938, 1994.

[103] C. Madsen and C. Andersen. Optimal landmark selection for triangulation of robot position. *Journal of Robotics and Autonomous Systems*, 13(4):277–292, 1998.

[104] P. Maes. A bottom-up mechanism for behavior selection in an artificial creature. In J-A. Meyer and S. W. Wilson, editors, *From animals to Animats: Proceedings of the First International Conference on Simulation of Adaptive Behavior*. MIT Press, 1991.

[105] M. C. Martin. *The Simulated Evolution of Robot Perception*. PhD thesis, Carnegie Mellon University, 2001. Technical Report CMU-RI-TR-01-32.

[106] M. C. Martin and H. P. Moravec. Robot evidence grids. Technical report, CMU Robotics Institute, Department of Computer Science, Chapel Hill, NC, USA, 1996. CMU-RI-TR-96-06.

[107] A. Martinoli. *Swarm Intelligence in Autonomous Collective Robotics: From Tools to the Analysis and Synthesis of Distributed Collective Strategies*. PhD thesis, DI-EPFL, Lausanne, Switzerland, 1999.

[108] A. Martinoli and F. Mondada. Collective and cooperative group behaviours: Biologically inspired experiments in robotics. In *Proceedings of the Fourth Symposium on Experimental Robotics ISER-95*, 1995.

[109] M. Mataric. A distributed model for mobile robot environment-learning and navigation. Master's thesis, MIT Artificial Intelligence Laboratory, 1990.

[110] E. W. Menzel. Group behavior in young chimpanzees: responsiveness to cumulative novel changes in a large outdoor enclosure. *J Comp Physiol Psychol*, 74:46–51, 1971.

[111] O. Michel. Webots: a powerful realistic mobile robots simulator. In *Proceeding of the Second International Workshop on RoboCup*. Springer-Verlag, 1998.

[112] O. Michel. Khepera simulator package. http://diwww.epfl.ch/ lami/ team/ michel/ khep-sim/, 2000.

[113] O. Michel. Professional mobile robot simulation. *International Journal of Advanced Robotic Systems*, 1(1):39–42, 2004.

[114] N. Minar, R. Burkhart, C. Langton, and M. Askenazi. The Swarm simulation system: a toolkit for building multi-agent simulations. Technical report, Santa Fe Institute, 1996. Working Paper 96-06-042.

[115] T. M. Mitchell. *Machine learning*. McGraw Hill, 1997.

[116] M. Montemerlo et al. Stanford racing team entry in the 2005 DARPA grand challenge. http://www.darpa.mil/ grandchallenge05/ TechPapers/ Stanford.pdf, 2006.

[117] H. P. Moravec. Sensor fusion in certainty grids for mobile robots. *AI Magazine*, pages 61–74, 1988.

[118] H. P. Moravec. The Stanford Cart and the CMU Rover. In *Proceedings of the IEEE*, volume 71, page 1983, 872–884.

[119] H. P. Moravec and M. Blackwell. Learning sensor models for evidence grids. Technical report, CMU Robotics Institute, Department of Computer Science, Chapel Hill, NC, USA, 1993. Annual Research Review, 1991.

[120] H. P. Moravec and A. Elfes. High resolution maps from wide angle sonar. In *Proceedings of the IEEE International Conference on Robotics and Automation, (St. Louis, MO)*, pages 116–121, 1985.

[121] D. Murray and C. Jennings. Stereo vision based mapping and navigation for mobile robots. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 1694–1699, 1997.

[122] L. Nadel and H. Eichenbaum. Introduction to the special issue on place cells. *Hippocampus*, 1999.

[123] U. Nehmzow and T. Smithers. Mapbuilding using self-organising networks in „Really Useful Robots". In J-A. Meyer and S. W. Wilson, editors, *First International Conference on Simulation of Adaptive Behavior*, 1991.

[124] A. Newell and H. A. Simon. Computer science as empirical enquiry: symbols and search. *Communications of the ACM*, 19(3):113–126, 1976.

[125] A. E. Nicholson and J. M. Brady. The data association problem when monitoring robot vehicles using dynamic belief networks. In *ECAI 92: 10th European Conference on Artificial Intelligence Proceedings*, pages 689–693. Wiley, 1992.

[126] N. J. Nilsson. Shakey the robot. Technical report, AI Center, SRI International, 1984.

[127] S. Nolfi, D. Floreano, O. Miglino, and F. Mondada. How to evolve autonomous robots: Different approaches in evolutionary robotics. In R. Brooks and P. Maes, editors, *Artificial Life IV*, pages 190–197. MIT Press/Bradford Books, 1994.

[128] S. Nolfi and D. Parisi. Evolving non-trivial behaviors on real robots: an autonomous robot that picks up objects. In *AI*IA*, pages 243–254, 1995.

[129] I. Nourbakhsh, J. Bobenage, S. Grange, R. Lutz, R. Meyer, and A. Soto. An affective mobile robot educator with a full-time job. *Artificial Intelligence*, 114(1–2):95–124, 1999.

[130] I. Nourbakhsh, R. Powers, and S. Birchfield. Dervish an office-navigating robot. *AI Magazine*, 16(2):53–60, 1995.

[131] J. Nyékyné G. et al. *Java 2 útikalauz programozóknak*. ELTE TTK Hallgatói alapítvány, 1999.

[132] J. Nyékyné G. et al. *Programozási nyelvek*. Kiskapu, 2003.

[133] D. S. Olton and R. J. Samuelson. Remembrance of places past: spatial memory in rats. *J. Exp. Psych. Anim. Behav. Proc.*, 2:97–116, 1976.

[134] J. B. Oommen, S. S. Iyengar, N. S. V. Rao, and R. L. Kashyap. Robot navigation in unknown terrains using learned visibility graphs. part I: The disjoint convex obstacle case. *IEEE J. of Robot. & Autom.*, 3(6):672–681, 1987.

[135] C. Papadimitriou. *Számítási bonyolultság*. Novadat Bt., 1999.

[136] H. Van Dyke Parunak and S. A. Brueckner. Entropy and self-organization in multi-agent systems. In *Proceedings of the International Conference on Autonomous Agents*, pages 124–130, 2001.

[137] H. Van Dyke Parunak, S. A. Brueckner, J. A. Sauter, and R. Matthews. Global convergence of local agent behaviors. In *AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multi-agent systems*, pages 305–312, New York, NY, USA, 2005. ACM Press.

[138] Cs. Pléh. *Bevezetés a megismeréstudományba*. TypoTex Elektronikus Kiadó, 1998.

[139] S. Quinlan and O. Khatib. Elastic bands: Connecting path planning and control. In *Proceedings of IEEE Int. Conference on Robotics and Automation*, pages 802–807, 1993.

[140] V. Ramos and F. Almeida. Artificial ant colonies in digital image habitats - a mass behaviour effect study on pattern recognition. In M. Dorigo, M. Middendoff, and T. Stützle, editors, *Proc. of ANTS'2000 - 2nd Int. Workshop on Ant Algorithms (From Ant Colonies to Artificial Ants)*, pages 113–116, 2000.

[141] V. Ramos, C. Fernandes, and A. C. Rosa. Social cognitive maps, swarm perception and distributed search on dynamic landscapes. *Brains, Minds & Media, Journal of New Media in Neural and Cognitive Science*, 2005.

[142] T. S. Ray. Evolution and optimization of digital organisms. *Scientific Excellence in Supercomputing: The IBM 1990 Contest Prize Papers*, pages 489–531, 1991.

[143] T. S. Ray. Evolution and complexity. *Complexity: Metaphors, models and reality*, 1994.

[144] S. F. Reardon and D. O'Sullivan. Measures of spatial segregation. *Sociological Methodology*, 34:121–162, 2004.

[145] D. Redish. *Beyond the Cognitive Map*. PhD thesis, School of Computer Science, Carnegie Mellon University, 1997. CMU-CS-97-166.

[146] C. W. Reynolds. Flocks, herds, and schools: A distributed behavioral model. *Computer Graphics*, 21(4):25–34, 1987.

[147] Robocup@home rules. http://www.ai.rug.nl/robocupathome/documents/AtHomeRules.pdf, 2006.

[148] M. Rosencrantz, G. Gordon, and S. Thrun. Decentralized sensor fusion with distributed particle filters. In *Proceedings of the Conference on Uncertainty in AI (UAI)*, Acapulco, Mexico, 2003.

[149] P. L. Rosin and G. A. West. Segmentation of edges into lines and arcs. *Image and Vision Computing*, 7(2):109–114, 1989.

[150] S. Russell and P. Norvig. *Mesterséges intelligencia modern megközelítésben*. Panem Kiadó, Budapest, 1999.

[151] L. Rühl. *Csillagászati navigáció*. Műszaki Könyvkiadó, 1970.

[152] H. A. Samani, A. Abdollahi, H. Ostadi, and S. Z. Rad. Design and development of a comprehensive omni directional soccer player robot. *International Journal of Advanced Robotic Systems*, 1(3):191–200, 2004.

[153] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 1948.

[154] A.W. Siegel and S. White. The development of spatial representations of large-scale environments. In H. Reese, editor, *Advances in Child Development and Behavior*, volume 10, pages 9–55. Academic Press, 1975.

[155] R. Siegwart and I. Nourbakhsh. *Introduction to Autonomous Mobile Robots*. A Bradford Book, The MIT Press, 2004.

[156] R. Simmons. The curvature-velocity method for local obstacle avoidance. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pages 3375–3382, 1996.

[157] H. A. Simon. *The Sciences of the Artificial*. The MIT Press, 1996.

[158] K. Sims. Evolving 3D morphology and behavior by competition. In *Fourth International Workshop on the Synthesis and Simulation of Living Systems*. MIT Press, 1994.

[159] K. Sims. Evolving virtual creatures. In *SIGGRAPH 94*, pages 15–22. ACM Press, 1994.

[160] E. Sklar, S. Parsons, and P. Stone. RoboCup in higher education: A preliminary report. *Journal on Educational Resources in Computing (JERIC)*, 4(2), 2004. Special issue on robotics in undergraduate education. Part 1.

[161] N. H. Sleumer and N. Tschichold-Gürman. Exact cell decomposition of arrangements used for path planning in robotics. Technical report, ETH Zürich, Institute of Theoretical Computer Science, 1999. TR-329.

[162] R. Smith, M. Self, and P. Cheeseman. Estimating uncertain spatial relationships in robotics. In I. J. Cox and G. T. Wilfong, editors, *Autonomous Robot Vehicles*, pages 167–193. Springer-Verlag, 1990.

[163] P. Stepan, M. Kulich, and L. Preucil. Robust data fusion with occupancy grid. *IEEE Transactions on Systems, Man and Cybernetics, Part C,*, 35(1):106–115, 2005.

[164] M. Stilman and J. Kuffner. Navigation among movable obstacles: Real-time reasoning in complex environments. In *Proceedings of the 2004 IEEE International Conference on Humanoid Robotics (Humanoids'04)*, 2004.

[165] P. Stone. *Layered Learning in Multiagent Systems: A Winning Approach to Robotic Soccer*. MIT Press, 2000.

[166] R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. Bradford Book, MIT Press, 1998.

[167] R. Szabó. Combining metric and topological navigation of simulated robots. *Acta Cybernetica*, 17(2):401–417, 2005.

[168] R. Szabó. Neural network controlling architectures in autonomous agents. In *Agents Everywhere – Proceedings of the First Hungarian National Conference on Agent Based Computing*, pages 99–109, 1999.

[169] R. Szabó. *Mobil robotok szimulációja*. Eötvös Kiadó, 2001.

[170] R. Szabó. Robotfutball. In Gy. Kampis and L. Ropolyi, editors, *Evolúció és megismerés, A 9. Magyar Kognitív Tudományi Konferencia Előadásai*, pages 399–414. Typotex Kft., 2001.

[171] R. Szabó. Navigation of simulated mobile robots in the Webots environment. *Periodica Polytechnica — Electrical Engineering*, 47(I-II):149–163, 2003.

[172] R. Szabó. A robotok navigációjának nehézségei. *Élet és tudomány*, LIX(47):1485–1487, 2004.

[173] R. Szabó. Topological navigation of simulated robots using occupancy grid. *International Journal of Advanced Robotic Systems*, 1(3):201–206, 2004.

[174] R. Szabó. Robotika és szimuláció - miért nehéz robotnak lenni? *Élet és tudomány*, LX(25):780–782, 2005.

[175] R. Szabó. A foglaltsági háló és már térképépítési stratégiák. In E. Kubinyi and Á. Miklósi, editors, *Megismerésünk korlátai*, pages 135–145. Gondolat Kiadó, 2006.

[176] R. Szabó. Occupancy grid based robot navigation with sonar and camera. In *Proceedings of CSCS'2006, The Fifth International Conference of PhD Students in Computer Science,University of Szeged*, 2006.

[177] R. Szabó and T. Vajna. Gép-ész – robotfoci világbajnokság Seattle-ben. *Heti Világgazdaság*, XXIII(34):82–83, 2001.

[178] S. Székely, I. Havas, and V. Csányi. How paradise fish(macropodus opercularis) explores a chessboard? *Acta Biol. Hung. Acad. Sci. Acad. Sci.*, 29:401–406, 1978.

[179] B. Takács and A. Lőrincz. Independent component analysis forms place cells in realistic robot simulations. *Neurocomputing*, 69:1249–1252, 2006.

[180] H. Theil. *Statistical decomposition analysis*. North-Holland Publishing Company, 1972.

[181] G. Theraulaz, J. Gautrais, S. Camazine, and J-L. Deneubourg. The formation of spatial patterns in social insects: from simple behaviours to complex structures. *Philos Transact R. Soc. Lond.*, 361(1807):1263–1282, 2003.

[182] S. Thrun. Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, 99(1):21–71, 1998.

[183] S. Thrun. Robotic mapping: A survey. In G. Lakemeyer and B. Nebel, editors, *Exploring Artificial Intelligence in the New Millenium*. Morgan Kaufmann, 2002.

[184] S. Thrun, A. Bücken, W. Burgard, D. Fox, T. Fröhlinghaus, D. Henning, T. Hofmann, M. Krell, and T. Schmidt. Map learning and high-speed navigation in RHINO. In D. Kortenkamp, R.P. Bonasso, and R Murphy, editors, *AI-based Mobile Robots: Case Studies of Successful Robot Systems*. MIT Press, 1998.

[185] R. Tobias and C. Hofmann. Evaluation of free Java-libraries for social-scientific agent based simulation. *Journal of Artificial Societies and Social Simulation*, 7(1), 2004.

[186] E. C. Tolman. *Purposive behavior in animals and men*. Appleton-Century-Crofts, New York, 1932.

[187] K. Tombre, C. Ah-Soon, P. Dosch, G. Masini, and S. Tabbone. Stable and robust vectorization: How to make the right choices. *Lecture Notes in Computer Science*, 1941:3–17, 2000.

[188] K. Tombre, C. Ah-Soon, Ph. Dosch, A. Habed, and G. Masini. Stable, robust and off-the-shelf methods for graphics recognition. In *Proceedings of the 14th International Conference on Pattern Recognition, Brisbane (Australia)*, pages 406–408, August 1998.

[189] D. S. Touretzky, H. S. Wan, and A. D. Redish. Neural representation of space in rats and robots. In J. M. Zurada and R. J. Marks, editors, *Computational Intelligence: Imitating Life*. IEEE Press, 1994.

[190] O. Trullier and J.-A. Meyer. Biomimetic navigation models and strategies in animats. *AI Communications*, 10(2):79–92, 1997.

[191] A. M. Turing. Computing machinery and intelligence. *Mind*, 59:433–460, 1950.

[192] I. Ulrich and I. R. Nourbakhsh. Appearance-based obstacle detection with monocular color vision. In *AAAI/IAAI*, pages 866–871, 2000.

[193] K. Wall and P.-E. Danielsson. A fast sequential method for polygonal approximation of digitized curves. *Computer Vision, Graphics and Image Processing*, 28:220–227, 1984.

[194] C. Wang and C. Thorpe. Simultaneous localization and mapping with detection and tracking of moving objects. In *Proceeding of the IEEE International Conference on Robotics & Automation (ICRA)*, 2002.

[195] Q. H. Wang, T. Ivanov, and P. Aarabi. Acoustic robot navigation using distributed microphone arrays. *Information Fusion (Special Issue on Robust Speech Processing)*, 5(2):131–140, 2004.

[196] M. Wilson, C. Melhuish, A. Sendova-Franks, and S. Scholes. Algorithms for building annular structures with minimalist robots inspired by brood sorting in ant colonies. *Auton. Robots*, 17(2-3):115–136, 2004.

[197] S. W. Wilson. The animat path to AI. In *Proceedings of the first international conference on simulation of adaptive behavior on From animals to animats*, pages 15–21, Cambridge, MA, USA, 1990. MIT Press.

[198] M. Wooldridge. Agent-based computing. *Interoperable Communication Networks*, 1(1):71–97, 1998.

[199] M. Wooldridge and N. R. Jennings. Intelligent agents: Theory and practice. *Knowledge Engineering Review*, 10(2):115–152, 1995.

[200] G. Wyeth. Hunt and gather robotics. In *International Conference on Field and Service Robotics*, pages 334–341, 1997.

[201] F. Zambonelli and H. Van Dyke Parunak. Signs of a revolution in computer science and software engineering. In *ESAW*, pages 13–28, 2002.