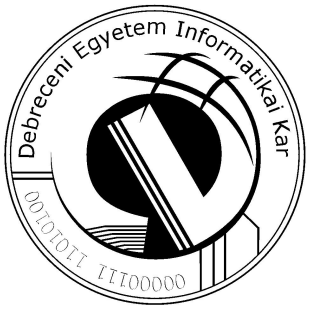


Algoritmusok és implementációk a swarm intelligencia területén

Doktori (PhD) értekezés

Bolla Kálmán Milán
Témavezető: Dr. Fazekas Gábor

Debreceni Egyetem
Természettudományi Doktori Tanács
Informatikai Tudományok Doktori Iskola
Debrecen, 2015.

Ezen értekezést a Debreceni Egyetem Természettudományi Doktori Tanács Informatikai Tudományok Doktori Iskola Alkalmazott információtechnológia és elméleti háttere program keretében készítettem a Debreceni Egyetem természettudományi doktori (PhD) fokozatának elnyerése céljából.
Debrecen, 2015. augusztus 10.

.....
Bolla Kálmán Milán
jelölt

Tanúsítom, hogy Bolla Kálmán Milán doktorjelölt 2009 - 2015 között a fent megnevezett Doktori Iskola Alkalmazott információtechnológia és elméleti háttere programjának keretében irányításommal végezte munkáját. Az értekezésben foglalt eredményekhez a jelölt önálló alkotó tevékenységével meghatározóan hozzájárult. Az értekezés elfogadását javasolom.
Debrecen, 2015. augusztus 10.

.....
Dr. Fazekas Gábor
témavezető

Algoritmusok és implementációk a swarm intelligencia területén

Értekezés a doktori (PhD) fokozat megszerzése érdekében
az informatika tudományágban

Írta: Bolla Kálmán Milán okleveles programtervező informatikus

Készült a Debreceni Egyetem Informatikai Tudományok doktori iskolája
Alkalmazott információtechnológia és elméleti háttere program keretében

Témavezető: Dr. Fazekas Gábor

A doktori szigorlati bizottság:

elnök:	Dr. Végh János	.....
tagok:	Dr. Hajdu András	.....
	Dr. Porkoláb Zoltán	.....

A doktori szigorlat időpontja: 2014. március 28.

Az értekezés bírálói:

	Dr.	.....
	Dr.	.....

A bírálóbizottság:

elnök:	Dr.	.....
tagok:	Dr.	.....
	Dr.	.....
	Dr.	.....
	Dr.	.....

Az értekezés védésének időpontja: 2015. .....

Tartalomjegyzék

Köszönetnyilvánítás

Szeretném megköszönni témavezetőmnek, Dr. Fazekas Gábornak és kollégámnak, Dr. Kovács Tamásnak a segítőkész támogatásukat és szakmai segítségüket.

Továbbá hálás vagyok családom türelméért és támogatásért. Nélkülük nem jöhetett volna létre az alábbi mű.

Bevezetés

Disszertációmat három fejezetre bontottam, amely időrendi sorrendben mutatja be doktori tanulmányaim alatt végzett kutatási munkámat. Minden fejezetet szakirodalmi áttekintéssel kezdek és bemutatom a témában releváns publikációk eredményeit. Az adott fejezet bevezetése után saját kutatási munkámat mutatom be, illetve amennyiben szükséges, a témához szorosan kapcsolódó korábbi eredményeket is részletezem. Kutatási eredményeim igazolásához és az algoritmusok helyes működésének bizonyításához kísérleteket és szimulációkat használtam fel.

Az 1. fejezetben bemutatásra kerül egy saját fejlesztésű, képfeldolgozáson alapuló eljárás, amely egy lehetséges megoldást ad a robottársak felismerésének problémájára mobil robot swarm-okon belül. A kidolgozott algoritmust felhasználhatjuk az egyes robotok azonosítására, valamint lehetséges a megfigyelőtől való távolság becslése is. Az eljárást egy kutatási feladatok végrehajtására is alkalmas autonóm robotra implementáltam és kísérletek segítségével bizonyítottam eljárásom működésének helyességét.

A 2. fejezetben swarm intelligencia alapfeladatokat mutatok be, ezeken belül is a gyülekezés problémáját emeltem ki elsősorban. Szakirodalmi áttekintés részben bemutatom és osztályozom a gyülekezési algoritmusokkal kapcsolatos legrelevánsabb eredményeket, a fejezet további részében pedig a témában folytatott két kutatási eredményem írom le. Az algoritmusok helyességét az általam fejlesztett szimulációs szoftverrel ellenőriztem, valamint összehasonlítottam az új megoldásokat a korábbi eredményekkel is.

A 3. fejezetben beltéri navigációt segítő lokalizációs eljárásomat mutatom be. E fejezet bevezetésében részletesen leírom miért nem triviá-

lis feladat a beltéri lokalizáció és részletezem a problémára eddig használt megoldásokat. A bevezetés után kerül bemutatásra saját, vizuális érzékelőt használó megoldásom, amely működésének helyességét ebben az esetben is kísérletekkel ellenőriztem.

1. fejezet

Társak felismerése és azonosítása

1.1. Bevezetés

Autonóm mobil robotoknak az esetek túlnyomó részében olyan környezetben kell feladatokat végrehajtani, ahol különböző objektumok, terepakadályok, valamint más ágensek találhatóak. Swarm intelligencia feladatok végrehajtásakor feltételezzük, hogy a swarm-ban található robotok felismerik a társakat, így elkülönítve őket a tereptárgyaktól vagy esetleg más swarmhoz tartozó egyedektől. Egy lehetséges megoldás, ha a robotok fel vannak szerelve kamerával (vizuális érzékelő rendszerrel), amely segítségével képesek közvetlen környezetüket érzékelni, társaikat felismerni és lokalizálni. Számos megoldást találhatunk az irodalomban [1, 2, 3, 4, 5, 6, 7], amely kameraképek alapján alakzatok vagy színek felismerésével határozza meg a környezetben dolgozó robotokat és azok elhelyezkedését. A továbbiakban az általam relevánsnak tartott munkákat foglalom össze röviden.

Rekleitis és társai [1] cikkében egy terület felderítési algoritmust mutatnak be két együttműködő robot felhasználásával. Feltételezik, hogy a két robot egymást folyamatosan nyomon tudja követni (ezt úgy valósítják meg, hogy amíg az egyik mozog a másik álló helyzetben marad), tehát

egymás látótávolságán belül helyezkednek el. Végrehajtás során a másik robot pozíciója fog segíteni a pontosabb térkép elkészítésében és a saját pozíció megállapításában egyaránt. A másik robot relatív pozícióját egy „robot tracker" érzékelővel határozza meg, ami egy vizuális érzékelőn alapuló megoldás. Mindkét robot fel van szerelve ezzel az érzékelővel, amelynek feladata a robotokra felszerelt henger alakú jelzés érzékelése és értelmezése. A fehér alapú hengeren körbefut egy sötét színű csík spirálisan, mely felett és alatt egy-egy fekete csík helyezkedik el. A mintázat lényege, hogy az alsó és felső jelzés között körbefutó spirális részből megmondható a robot relatív irányultsága. Így az általuk javasolt algoritmus azon túl, hogy a rögzített képen látható robotokat meghatározza, annak relatív pozícióját is meg tudja becsülni.

Az előző esetben minden robot saját érzékelővel rendelkezett, viszont találhatunk olyan megközelítést is, amikor a vizuális érzékelő fixen helyezkedik el a környezeten belül és teljesen belátja a robotok által használt munkaterületet [6]. Így a robotok egy intelligens térben fogják végrehajtani a feladataikat. Engedy és Horváth munkájában [6] a kameraérzékelő és a hozzárendelt feldolgozó egység feladata a robotok és akadályok felismerése és követése a dinamikusan változó környezetben. Az egész rendszernek csak a belátható környezetről van tudomása, a nem belátható részekről nincs semmilyen információja. Az érzékelő a sík területhez képest felülnézetből, 320×240-es felbontással és másodpercenként 15 képkockával rögzít képeket. Itt is, mint az előző esetben a környezet jellemző objektumai (robotok és akadályok) jelzésekkel vannak ellátva. Amennyiben egy objektumot a rendszer nem ismer fel, azt automatikusan akadálynak tekinti. A változások érzékelésére a következő algoritmust használja, ahol feltételezi a környezet statikus megvilágítási viszonyait: az ismert háttérből kivonja az aktuálisan rögzített jelenetet, így a különbség által megkapja a képen található változásokat. Utófeldolgozásként morfológiai zárással és nyitással törli a hibákat a különbségképről. Az érzékelendő robotokra, azonosítás céljából különböző alakzatok lettek elhelyezve, melyek egyértelműen azonosítanak egy egyedet. Az algoritmus meghatározza az egyes alakzatok körvonalait, majd veszi azok Fourier-transzformáltját annak érdekében, hogy elforgatás invariáns mintaillesztést lehessen végrehajtani. Végül a feldolgozott

alakzatokat összehasonlítja a tárolt adatokkal, így megkapja a robotok és akadályok aktuális pozícióját a térben.

Társfelismerési algoritmusok fontos szerepet töltenek be a robotfoci (RoboCup) versenyeken is [2, 3, 5, 7]. A RoboCup versenyt 1997-ben indították abból a célból, hogy a robotika és a mesterséges intelligencia kutatásokat népszerűsítsék és az elért eredményeket gyakorlatban is be tudják mutatni egy évenkénti megrendezésű verseny keretein belül. RoboCup esetében a robotoknak csapatként kell együttműködniük, hasonlóan mint egy robot swarm esetében is. Játék közben a csapattársakat és az ellenfél játékosait fel kell tudni ismeri annak érdekében, hogy a megfelelő taktikát válasszák és saját ismereteiket és stratégiájukat fejlesszék. A RoboCup-ban szerepeltetett robotok (pl.: Nao, Sony Aibo) legtöbbször rendelkeznek kamera érzékelővel, mely kis felbontású, de színes képeket képes rögzíteni. A különböző csapatokba tartozó robotok általában színnel vannak megkülönböztetve egymástól, ezért sok algoritmus alapja a színek érzékelése a rögzített képen. Ezek közül is Wilking és Röfer [2], valamint Laue és Röfer [5] munkáját emelném ki, mindkét cikkben Sony Aibo robotokat használtak.

Wilking és Röfer cikkében [2] azt tűzték ki célul, hogy a robot különböző részein elhelyezett markerek segítségével meg tudják mondani a robot relatív pozícióját a megfigyelő szemszögéből egyetlen kamerakép alapján. Feltételezik, hogy a markerek egyedi színűek, tehát a környezet más színével nem összetéveszthetők. Algoritmusuk két részre, előfeldolgozásra és a pozíció felismerésére bontható. Az előfeldolgozás lépésben szín alapú szegmentációval elválasztja a hasznosnak ítélt objektumpontokat, majd kontúrkövetéssel megállapítja a felismert felületek határvonalait. A pozíció becsléséhez az előfeldolgozási lépésből származó adatok alapján attribútumokat generál, majd végrehajtja az így kapott attribútumok klaszterezését (tree learning algoritmussal). Végül a robotról eltárolt, 180 fokban elforgatott attribútum értékek alapján kapjuk annak relatív pozícióját. Az eljárás segítségével ezen felül meg tudjuk becsülni a megfigyelőtől vett távolságot is, mivel a felületek méretei már előzetesen ismertek a robotok számára.

Egy másik felismerési eljárásban [5] egy alternatív megoldást alkalmaztak. Cél a társak és akadályok felismerése volt, illetve a társak távolságának meghatározása a robottól. A rögzített képen egy rácsot definiáltak,

ami a kép aljától a tetejéig folyamatosan sűrűsödik. Ezután csak a rácspontokat kell megvizsgálni, az előre definiált szín és egyéb szabályok szerint pedig el tudjuk dönteni, hogy az melyik keresendő objektumhoz tartozhat. Amennyiben egy objektumpontot találtunk, a rácson való elhelyezkedése alapján tudunk becslést adni annak távolságáról. Társak érzékelésén túl az akadályok felismerésére is javasoltak egy eljárást, ami meglehetősen RoboCup specifikus, ezért itt nem kerül tárgyalásra.

Könnyen belátható, hogy a vizuális érzékelőn alapuló objektum-felismerési, illetve mintaillesztő képfeldolgozási eljárások számításigényesek. Az általam bemutatott és tanulmányozott megoldásokból két lehetséges irány tárul elénk: magán a roboton végezzük el az érzékelőből származó információk feldolgozását vagy egy külső rendszert veszünk igénybe, amivel folyamatosan kapcsolatban vannak a robotok. Utóbbi esetben a robotok csak végrehajtják a jelenlegi és korábbi érzékelések alapján a parancsokat, így nem viselkednek autonóm módon. Amennyiben a feldolgozást csak a robot végzi lassabb számítási teljesítmény áll rendelkezésünkre, ezért ilyenkor a rögzített kép felbontását érdemes csökkenteni. Célom egy olyan robotfelismerési eljárás tervezése és megvalósítása volt, ami magán a roboton fut, továbbá szerényebb képességű hardveren is képes, legalább VGA (640×480) felbontású képekkel a dinamikusan változó környezet érzékelésére.

1.2. Társfelismerés

Az általam kifejlesztett társfelismerő eljárás képfeldolgozáson alapuló módszer, ezért feltételezzük, hogy a swarm-ban található robotok fel vannak szerelve képalkotó eszközzel. Kamera által rögzített szürkeárnyalatos kép alapján kell az egyedeknek eldönteni, hogy a közvetlen környezetükben hol helyezkednek el a társak, valamint a környezet további objektumai.

Az eljárás alapja, hogy a swarm egyedeit egy ismétlődő mintázattal jelöljük meg. Ennek az ismétlődő mintázatnak jól kell látszódnia az algoritmus bemeneteként használt rögzített szürkeárnyalatos képen, ezért a mintázat fekete és fehér színek ismétlődéséből áll. Ismétlődő mintázatok detektálására az irodalomban számos megoldást találhatunk [8, 9, 10, 11,

12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25], amelyek legtöbbje nem használható a robot swarm-okban használatos szerény számítási képességekkel és csekély memóriával felszerelt egyedek esetében. Egy lehetséges megoldás, ha a robotok a rögzített képet elküldik egy külső munkaállomásnak feldolgozásra, ezzel megkerülve az egyedek gyenge számítási képességből adódó problémáját. Mivel a kifejlesztett algoritmus egy konkrét, fizikai roboton implementálásra került, olyan megoldást kellett találni a társfelismerés problémájára, ami akár egy 500 MHz-es mikrovezérlőn is képes működni.

A kamera által rögzített szürkeárnyalatos jelenetet ($n \times m$ felbontású) oszloponként mintavételezzük, mivel feltételezzük, hogy a robot swarm tagjai sík terepen dolgoznak. Így a minta esetleges elforgatottjaival nem kell foglalkozni, kevesebb számítást kell végrehajtani. Az általam kifejlesztett társfelismerő eljárás a Fourier-analízisen alapszik a mintázat jellege miatt. Amennyiben egy ismétlődő mintázatot mintavételezünk és azt a frekvenciatartományban elemezzük, egy jól kivehető lokális maximumot láthatunk. E lokális maximum megléte, illetve nemléte alapján el tudjuk dönteni, hogy az adott jeleneten látunk e társat vagy sem. Továbbá a kapott lokális maximum helye függ a mintázat által generált frekvenciától, ha a mintázatot távolabb vagy közelebb helyezzük el a megfigyelőtől, attól függően a lokális maximumot magas, illetve az alacsony frekvenciatartományban fogjuk érzékelni.

1.2.1. Mintavételezés

Az általam alkalmazott mintafelismerésben nagy hangsúlyt kap a rögzített jelenet mintavételezése. Belátható, hogy egy $n \times m$-es felbontású kép esetén a teljes kép elemzése időigényes feladat, esetemben pedig egy jól meghatározott mintázatot kell megtalálni. A társfelismerési algoritmus alapjául szolgáló gyors Fourier-transzformáció (FFT) időbonyolultsága $n \cdot \log(n)$, amely bemenetként a mintavételezett kép oszlopait kapja meg. Amennyiben a teljes jelenetet feldolgoznánk, $m \cdot n \cdot \log(n)$ időbonyolultsággal kellene számolnunk, mind a legkedvezőbb, mind a legrosszabb esetben egyaránt. Legkedvezőbb esetre (amikor nem található meg a mintázat a rögzített ké-

pen) egy lehetséges megoldás, ha bevezetünk egy mintavételezési konstanst (c), mely segítségével felbontástól függetlenül tudunk oszloponként mintavételezni, ami azért lehetséges, mert a mintázat mérete egyenes arányban nő a rögzített kép felbontásának növelésével. Az így kapott legkedvezőbb mintavételezési eset $c \cdot n \cdot \log(n)$ időbonyolultságú lesz minden felbontás mellett, ezért csak az oszlopokban található képelemek számától (n) függ a legkedvezőbb eset időbonyolultsága. Természetesen, ha az egész képen az adott ismétlődő mintázat található, akkor a legrosszabb eset áll fenn, tehát a teljes digitális képet fel kell dolgozni, az algoritmusunk időbonyolultsága pedig $m \cdot n \cdot \log(n)$ lesz.

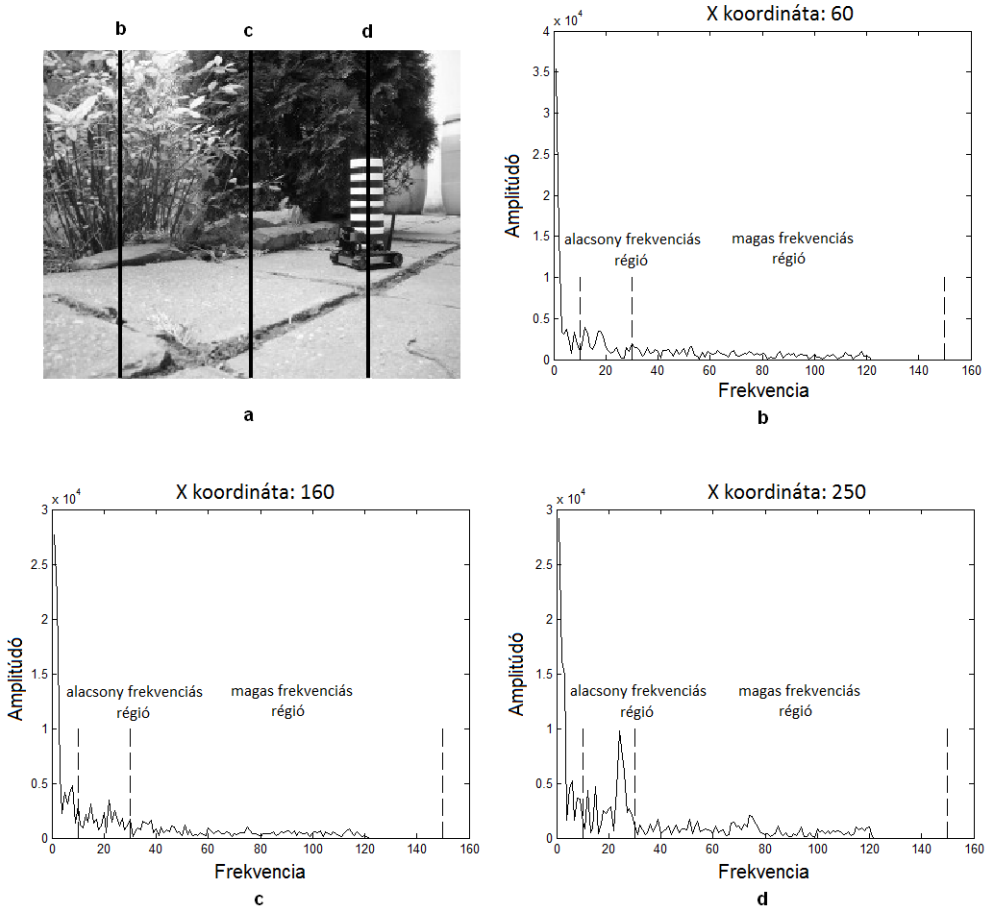
1.2.2. Egy oszlop vizsgálata

Társak azonosításának alapötlete az 1.1. ábrán látható. Minden egyes robotot a swarm-ban felszerelünk egy ismétlődő mintázattal, így azok felismerhetővé válnak a társak számára. A robotok által rögzített digitális képeket oszloponként mintavételezzük és az így kapott egydimenziós jel lesz a bemenete a gyors Fourier-transzformációnak. A Fourier-spektrumot a következőképpen számoljuk:

$$\bar{C} = |F(\bar{S})| \tag{1.1}$$

ahol $\bar{S}$ a mintavételezett oszlop pixel érték vektora, $\bar{C}$ pedig a Fourier-együtthatók vektora.

Az 1.1. ábrán látható három különböző mintavételezésből származó Fourier-spektrum, valamint az eredeti kép (1.1/a). 1.1/b és 1.1/c ábrákon látható egy-egy olyan mintavételezett oszlopnak a Fourier-spektruma, mely nem tartalmazza a mintázatot, viszont az 1.1/d ábrán látható egy jól kivehető lokális maximum. Ezt a lokális maximumot az ismétlődő mintázat generálta, mivel a kiválasztott oszlop a mintázat részét képzi.



1.1. ábra. Robot által rögzített szürkeárnyalatos kép (a) és a három mintavételezett oszlop Fourier-spektruma (b), (c), (d). (d)-n egy jól kivehető lokális maximumot láthatunk, amelyet az ismétlődő mintázat generált.

Legyen f_p a keresendő lokális maximum a frekvenciatartományban és A_p a hozzátartozó érték. Jól látható az ábrákon, hogy az alacsony frekvenciás régió értékei jóval magasabbak, mint a magas frekvenciás régióban található értékek és a keresendő lokális maximum, ezért az alacsony frekvenciás régiót nem vesszük figyelembe a mintakeresés során. Az így kapott további frekvenciatartományt két régióra bontjuk, egy alacsony (R_{low}) és egy magas frekvenciás (R_{high}) régióra, amelyekben külön keressük a lokális maximumokat. Erre a bontásra azért van szükség, mert ha a mintázat a magasabb frekvenciás régióban generál egy csúcsot, akkor az kisebb lehet, mint a R_{low} régióban található, nem a mintázat által generált érték. Az R_{low} és R_{high} régiók határait tapasztalati úton határoztam meg:

$$R_{low} = [f_{min}, f_{mid}[\tag{1.2}$$

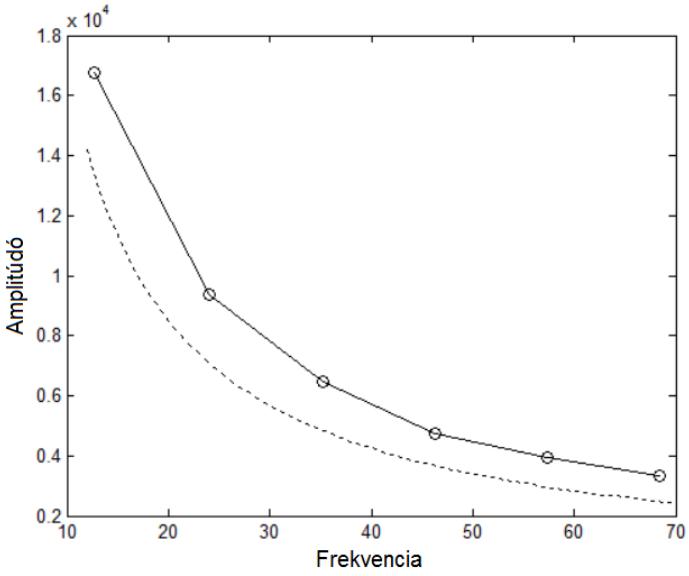
$$R_{high} = [f_{mid}, f_{max}] \tag{1.3}$$

ahol $f_{min} = 10$, $f_{mid} = 30$, $f_{max} = 150$ (mértékegység: Nyquist frekvencia / M egység).

A mintavételezés során elsődleges feladatunk meghatározni, hogy a mintavételezett oszlop áthalad-e az ismétlődő mintázaton vagy sem. A Fourier-spektrum elemzése során az előbb említett két régióban két lokális maximum értéket határozunk meg (A_{min}, A_{max}) egy küszöbfüggvény segítségével. Amennyiben olyan lokális maximumot találunk a két régió valamelyikében, ami a küszöbfüggvény által meghatározott értéknél magasabb értéket vesz fel, a mintavételezett oszlop a mintázat részének tekinthető. A küszöbfüggvényt szintén tapasztalati úton határoztam meg 50-től 300 cm-ig, 50 cm-es léptékkel készített referenciaképek alapján, az 1.2. ábrán található értékek segítségével. A kapott eredmények jól láthatóan egy hatványfüggvényre hasonlítanak, így a közelítő függvényt a következőképpen határoztam meg:

$$T(f) = 1.7E^5 \cdot \frac{1}{f} \tag{1.4}$$

ahol f a frekvencia, $T(f)$ pedig a küszöbfüggvény.



1.2. ábra. 50 cm-től 300 cm-ig, 50 centiméterenként megismételt mérések eredményei és a mintázatot kiválasztó küszöbfüggvény. A mért távolságok átlagát körrel, a tapasztalati úton meghatározott küszöbfüggvényt pedig szaggatott vonallal jelöltem az ábrán.

1.2.3. Digitális kép feldolgozása

Az előbbiekben meghatároztuk egy mintavételezett oszlop feldolgozását a frekvenciatartományban. Következő lépésként a teljes mintázat vagy mintázatok (attól függően, hogy hány robot található a képalkotó környezetében) meghatározása a cél a teljes digitális képen, úgy hogy semmilyen a priori információ nem áll a robotok rendelkezésére. Az előzetes tudás hiánya azt jelenti, hogy az aktuális jelenetet megelőző képeken talált mintázatok elhelyezkedése nem befolyásolja a jelenlegi kép feldolgozását.

A mintafelismerés során felmerülő hibák minimalizálása érdekében meg kellett határozni egy minimális és egy maximális távolságot, amely tartományban az algoritmus még megbízhatóan működik. Számos kísérlet után úgy állapítottam meg, hogy 50 cm-től 300 cm-ig bezárólag jól teljesít a társfelismerő eljárás. 300 cm-nél távolabb már egyre többször tapasztalhatunk hibákat, amely hibák elsősorban a VGA felbontásból és a kameraérzékelő minőségéből adódnak. A 300 cm-es határnál a mintázat egy VGA felbontású (640×480) képen mindössze 10 pixel-en jelenik meg horizontálisan. Minimális, 50 cm-es határra azért volt szükség, mert a mintázat fizikai méretei miatt nem látható be teljesen a rögzített képen.

A társfelismerő algoritmus három eljárásra lett szétbontva. Első, és egyik legfontosabb az *AnalyzeOneColumn* eljárás (1. algoritmus), ami egyetlen mintavételezett oszlopot dolgoz fel, kimenete a mintázat által generált frekvencia értéke. Amennyiben nem talál a feltételeknek megfelelő lokális maximumot az R_{low} és R_{high} régiókban, eredmény nélkül tér vissza.

Abban az esetben, ha az *AnalyzeOneColumn* eljárás eredménnyel tér vissza, a $WholePattern$ eljárásnak (2. algoritmus) lesz a feladata, hogy a mintázathoz tartozó további oszlopokat meghatározza. Működése során a kapott oszlop bal illetve jobbszomszédságában keresi oszloponként a mintázat határait. Kimenete a mintázat kezdeti (P_{start}) és vég oszlopának (P_{end}) x koordinátái a digitális képen.

Algoritmus 1 Mintázat keresése egy oszlopban

Eljárás AnalyzeOneColumn

1: **Input:** mintavételezendő oszlop indexe (x_0)
2: mintavételezés x_0 alapján és mentés $\bar{S}$-ba
3: $C(f)$ FFT spektrum számítása $\bar{S}$ alapján
4: lokális maximumok értékeinek ($A_{p_{low}}$ és $A_{p_{high}}$) és helyeinek ($f_{p_{low}}$ és $f_{p_{high}}$) meghatározása a két régióban (R_{low} és R_{high})
5: **if** $A_{p_{low}} > A_{p_{high}}$ **then**
6: **if** $A_{p_{low}} > T(f_{p_{low}})$ **then** $f_{result} = f_{p_{low}}$
7: **if** $A_{p_{high}} > T(f_{p_{high}})$ **then** $f_{result} = f_{p_{high}}$
8: **else**
9: **if** $A_{p_{high}} > T(f_{p_{high}})$ **then** $f_{result} = f_{p_{high}}$
10: **if** $A_{p_{low}} > T(f_{p_{low}})$ **then** $f_{result} = f_{p_{low}}$
11: **end if**
12: **if** $A_{p_{low}} \leq T(f_{p_{low}})$ **OR** $A_{p_{high}} \leq T(f_{p_{high}})$ **then**
13: f_{result} =nincs eredmény
14: **end if**
15: **Output:** detektált frekvencia (f_{result})

Végül az utolsó függvényben ($SearchForRobots$, 3. algoritmus) az előző két eljárás felhasználásával megvalósítjuk a korábban említett mintavételezési megoldást, ahol a mintavételezési konstans $c = 80$-nak lett beállítva. Az algoritmus végrehajtása során csak azokat a detektált mintákat tekintjük jó eredménynek, melyek egy minimális oszlopszámon jelennek meg (P_{min}), ahol $P_{min} = 10$-nek lett választva. Erre azért volt szükség, hogy az érzékelésből adódó hibákat kiküszöböljük.

Algoritmus 2 Teljes mintázat keresése

Eljárás WholePatternFound

Input: mintavételezendő oszlop indexe (x_0)
$f_{sum} = 0$
$left = 1$
while $x_0 - left > 0$ **AND** $PatternFoundAt(x_0 - left)$ eredménnyel tér vissza
    $f_{sum} = f_{sum} + f_p$
    $left = left + 1$
end while
$P_{start} = x_0 - left$
$right = 1$
while $x_0 + right <$ Kép szélessége **AND** $PatternFoundAt(x_0 + right)$ eredménnyel tér vissza
    $f_{sum} = f_{sum} + f_p$
    $right = right + 1$
end while
$P_{end} = x_0 + right$
$f_{mean} = f_{sum}/(P_{end} - P_{start})$
Output: frekvenciák átlaga (f_{mean}), mintázat kezdete (P_{start}) és vége (P_{end})

1.3. Társak azonosítása

A robotok azonosításához módosítani kell a meglévő ismétlődő mintázatot, ami végső soron egy vonalkódnak fog megfelelni. Az eredeti mintázat a méretkorlátok miatt 5 fekete csíkot tartalmazott, ezt a mintázatot kellett azonosíthatóvá tenni, így lett az egyik csík másfélszer akkora, mint a többi. Ezzel a módszerrel összesen 5 különböző robot azonosítható, viszont ha egyszerre kettőt változtatunk meg, akkor már 10 különböző lehetőséget kapunk.

Algoritmus 3 Teljes digitális kép feldolgozása

Eljárás SearchForRobots

Input: robot által rögzített kép
$Patterns$ és $Frequencies$ üres tömbök
$\Delta x = N/c$
$i = 1$
while $i \leq N$
  if $PatternFoundAt(x_i)$ eredménnyel tér vissza **then**
    $[P_{start}, P_{end}, f_{mean}] = WholePatternFound(x_i)$
    if $P_{end} - P_{start} > P_{min}$ **then**
      P_{start} és P_{end} hozzáadása $Patterns$ tömbhöz
      f_{mean} hozzáadása $Frequencies$ tömbhöz
    end if
    $i = i + \Delta x \cdot [(P_{end} - i/\Delta x) + 1]$
  else $i = i + \Delta x$
  end if
end while

Társfelismerés után rendelkezésünkre állnak azok az oszlopok, melyek a mintázatot tartalmazzák, viszont az azonosításhoz a mintázat pontos helyére lesz szükségünk. Ennek megoldására egy másik képfeldolgozási algoritmust kellett kifejleszteni.

A továbbiakban nem a frekvenciatartományban dolgozunk tovább, hanem képtartományban. Az irodalomban számos, ismétlődő mintázattal kapcsolatos képfeldolgozási eljárást találhatunk, a saját algoritmusom alapját Yalniz és Aksoy [25] munkája adta, amellyel a mintázat meglétét lehet vizsgálni, valamint lokalizálni a képen emberi beavatkozás nélkül. Teljes egészében nem lehetett átvenni a módszert, mivel egy 1000×1000 pixeles képen 90 percig fut az algoritmusuk egy mai modern nagyteljesítményű számítógépen. A mi esetünkben szerencsére rendelkezésre állnak bizonyos a priori információk a mintázatról, mivel tudjuk, hogy fekete és fehér vonalak (textúra primitívek) váltakozását keressük, valamint a fekete és fehér párok számát is előzetesen tudjuk. Továbbá nem foglalkozunk a mintázat

elforgatottjaival, mivel egyenes padlós környezetet feltételezünk. Ezen információk alapján már lehetővé válik egy gyengébb teljesítményű eszközre kialakítani egy azonosító algoritmust.

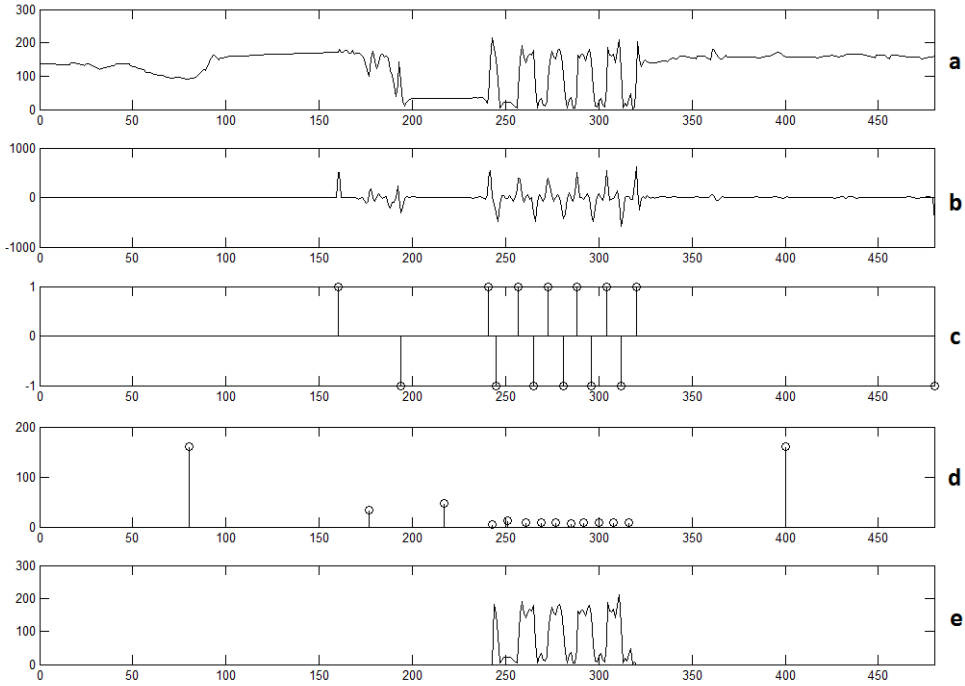
1.3.1. Mintázat lokalizációja

Az általam használt referencia cikkben [25] Laplacian of Gaussian (LoG) él-detektálást használnak, amit a teljesítmény szempontjából jobb Sobel operátorra cseréltem le, ezen belül is csak a vertikális irányú komponensre volt szükség a mintázat elhelyezkedése miatt. Munkám korábbi változatánál [26] LoG él-detektálást alkalmaztam, viszont akkor még az algoritmusomnak egy MATLAB-ban [27] implementált változata létezett csak.

Mivel a mintázat elég kontrasztos, így az élek lépcsős élként jelentkeznek a mintázatot tartalmazó oszlopban. Ebben az esetben a gradiens alapú Sobel él-detektálás használatával jól ki tudjuk emelni a képen található intenzitás változásokat. Az egyes oszlopokat külön fogjuk kezelni, mivel előfordulhat olyan szituáció, hogy két robot takarja egymást. Ilyenkor a társfelismerő eljárást alapból egynek tekintené a két egymás mellett található robotot, így az oszlopokat külön kell kezelnünk egymástól.

1.3. ábrán látható a mintavételezett oszlop intenzitás értékei, valamit a Sobel az él-detektálás eredménye (1.3/b ábra). Az él-kiemelés során a homogén régiókat elnyomtuk, a mintázathoz és a környezet egyéb objektumaihoz tartozó intenzitás változásokat pedig kiemeltük. Továbbiakban az él-detektálást egy küszöbölés követi, ahol két küszöbérték segítségével (lokális minimumok és maximumok külön) elválasztjuk az erős éleket a többi éltől, illetve hibás érzékeléstől (1.3/c ábra). Ezután a küszöbölt értékeket szegmensekre bontjuk és vizsgáljuk az egyes szegmensek szélesség tulajdonságát (1.3/d ábra), amely szükséges lesz a mintázat valós helyének meghatározásához, így csökkentve a helytelen azonosítások számát.

A felhasznált ismétlődő mintázat csúcsok és völgyek, textúra primitívek ismétlődéséből áll, ahol fix számú szegmens található. Az algoritmus először az összes lehetséges, 5 ismétlődést tartalmazó részt határozza meg a küszöbölt eredményeken, majd a szélesség tulajdonságok alapján dolgozik tovább. Minden lehetséges esetre kiszámolja a szélesség tulajdonságok



1.3. ábra. Mintázatot tartalmazó oszlopon végrehajtott azonosítás lépései, ahol a vizszintes tengelyen láthatóak a sorindexek, a függőleges tengelyen pedig a pixelek intenzitás értékei: (a) eredeti oszlop intezitás értékei, (b) Sobel él-detektálás eredménye, (c) él-detektálás utáni küszöbölés, (d) szegmensek szélesség tulajdonsága, (e) mintázat meghatározása a küszöbölt értékek és a szélesség tulajdonságok alapján.

alapján a szélességek varianciáját. Ahol a legkisebb lesz a variancia (az élek egymástól közel egyenlő távolságra helyezkednek el), ott határozzuk meg a mintázat pontos helyét. Ezzel a két feltétellel már megadható a minta helye a mintavételezett oszlopban. Az eljárás eredményét az 1.3/e ábra mutatja.

1.3.2. Azonosítás folyamata

Jelenlegi információk alapján már lehetővé válik a robotok azonosítása, ehhez mindössze a vastagabb fekete csík helyét kell meghatározni a mintázaton belül. Az eljárás közben megoldást kapunk a robotok egymás takarásának problémájára is, mivel a társfelismerés során kiválasztott oszlopokat egyesével dolgozzuk fel. Így az azonosítással egyúttal javíthatjuk a társfelismerés során felmerült problémákat is.

Algoritmus 4 Robotok azonosítása

Eljárás IdentifyRobots

1: **Input:** $Patterns$ tömb, ami tartalmazza a mintázatokat tartalmazó oszlopokat sorvektorként
2: Ids üres tömb
3: $N = Patterns$ tömb sorainak a száma
4: $i = 1$
5: **while** $i \leq N$
6: $p = Patterns[i]$
7: $sobel = p$-n Sobel él-detektálás végrehajtása
8: $thresholded = sobel$-n végrehajtott küszöbölés
9: $widths = thresholded$ tömben meghatározzuk a szegmensek hosszát
10: $thresholded$ és $widths$ tömbök alapján azonosítjuk a robotokat és az eredményeket tároljuk a Ids tömbben
11: $i = i + 1$
12: **end while**

1.4. Társak távolságának becslése

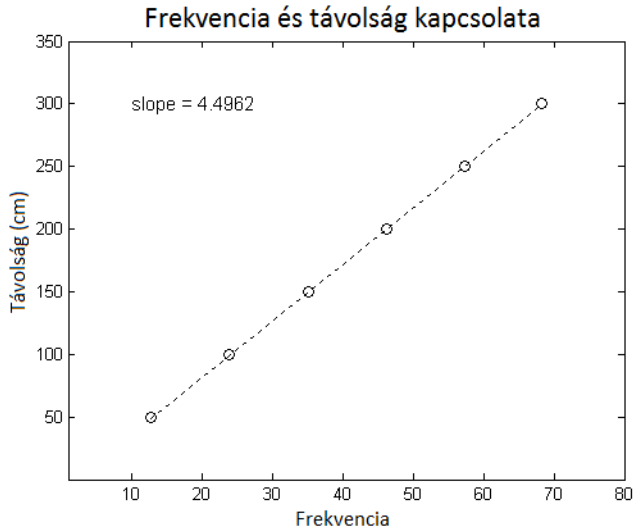
A Fourier-analízis egyik nagy előnye, hogy a mintázat által generált frekvencia alapján megbecsülhető a társak és a megfigyelő robot közötti távolság, mindez egyetlen rögzített jelenet alapján. Ennek oka, hogy ha a mintázatot a megfigyelőhöz közel helyezzük el, akkor a frekvenciatartományban létrejött lokális maximum az alacsony frekvenciás régióban keletkezik, viszont ha távol helyezzük el magasabb frekvenciát generál. A módszer alkalmazása jelentős plusz számítást nem igényel, mivel a detektált frekvencia már rendelkezésre áll. Pontosabb eredmény érdekében a teljes, felismert mintázat által generált frekvenciák átlagával számolunk (f_{mean}).

A frekvencia és a távolság közötti kapcsolat megállapítására kísérletekre volt szükség. A környezetben négy robotot helyeztem el: egy megfigyelő és három mintázattal ellátott robotot. Előbbi feladata az volt, hogy folyamatosan képeket rögzítsen a környezetéről és továbbítsa azokat egy külső számítási egységnek. Utóbbiakat pedig arra használtam, hogy meghatározott távolságra helyezzem el a megfigyelőtől, és ezáltal felmérjem a frekvencia és a távolság közötti kapcsolatot. Három különbözőképpen azonosítható mintázatot használtam a mérések során (1.1 táblázatban *robot1*, *robot2*, *robot3*-al jelöltem), amire azért volt szükség, hogy ne kapjunk olyan eredményt, ami csak az egyik mintázatra illeszkedik, viszont a többi esetben rossz becslést kapnánk. A kísérleteket 50 cm-től 300 cm-ig, 50 cm-es léptékkel végeztem, amiből az derült ki, hogy lineáris kapcsolat fedezhető fel a generált frekvencia és a távolság között. Mérések részletes eredményeit az 1.1. táblázat tartalmazza: az adott távolságban detektált frekvenciák (f_p) mérésenkénti átlagát, valamint a három mintázatnál kapott értékek átlagát.

	robot1	*robot2*	*robot1*	Átlag	Szórás
50 cm	13.1	12.5	12.6	12.7	0.3215
100 cm	24.7	22.6	23.5	23.9	1.0536
150 cm	36	34.6	34.7	35.1	0.7810
200 cm	48.9	45.3	44.6	46.3	2.3072
250 cm	58.9	56.6	56.7	57.4	1.3
300 cm	69	67.7	67.8	68.2	0.7234

1.1. táblázat. Frekvenciák alakulása a távolság függvényében.

Az 1.4 ábrán látható a távolság és frekvencia közötti kapcsolat. A mérföldköveknél kapott eredményeket az ábrán körökkel jelöltem, az illesztett közelítő egyenest pedig szaggatott vonallal, melynek meredeksége 4.4962.



1.4. ábra. A mért frekvencia és távolság lineáris kapcsolata, ahol a körök jelölik a kísérletek során mért eredményeket, a szaggatott vonal pedig az eredményekre illesztett egyenest jelöli.

A mérések alapján arra a megállításra jutottam, hogy a frekvenciából becsült távolságmérés ±10 cm-es pontossággal működik.

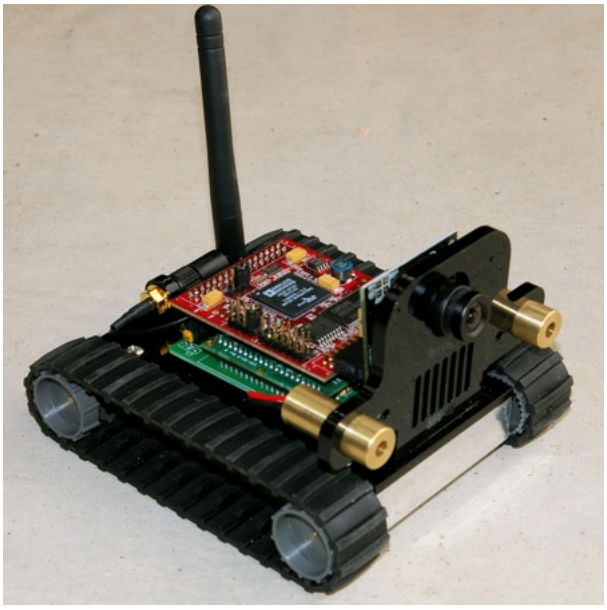
1.5. Algoritmus implementálása és működés autonóm mobil roboton

Annak érdekében, hogy az algoritmusom helyes működését bemutassam, egy konkrét roboton implementáltam a társfelismerő és azonosító algoritmusomat. A megvalósításhoz Surveyor által gyártott SRV-1 robotokat használtam, a robotokat pedig felszereltem a társfelismeréshez és azonosításhoz szükséges ismétlődő mintázattal.

1.5.1. Surveyor SRV-1 robot

A Surveyor cég gyártotta SRV-1 (1.5. ábra) mobil robot oktatási és kutatási célokra lett kifejlesztve, elsősorban felsőoktatási intézmények számára. Központi vezérlő egységként egy 500 MHz-es Analog Devices Blackfin BF537 processzor szolgál, valamint található a roboton egy Blackfin típusú kamera is, mely 160×128-tól 1280×1024 felbontásig képes színes képeket rögzíteni YUV422 formátumban. A Lantronix Matchport segítségével pedig vezeték nélküli kapcsolaton keresztül tud kommunikálni más eszközökkel.

Szoftveresen a robot teljesen nyílt forráskódon alapul, operációs rendszerként egy μLinux rendszer szolgál, amelyen a C nyelvű gyári program fut. A gyári program a robot bekapcsolása után telnet-en keresztül tud fogadni egyszerű parancsokat, amellyel a motorok irányíthatók és a szenzorok által rögzített értékek kiolvashatók. A robot alapértelmezetten egy állapotgépként üzemel, számos kliens alkalmazás található hozzá, amivel az érzékelők és a beavatkozók tesztelhetőek a Surveyor cég által meghatározott protokoll segítségével.



1.5. ábra. Surveyor SRV-1 robot.

Robot programozására alapvetően két lehetőségünk van: az egyik a gyártó által felkínált, kezdő programozóknak is könnyen elsajátítható picoC nyelv, ami alapvetően egy C alapú nyelv a robot speciális függvényeivel kibővítve. Hátránya, hogy nem lehet benne bonyolultabb programokat írni, így például saját függvényt definiálni sem. Másik lehetőség a gyári program módosítása, ami komolyabb programozási és mikrovezérős tudást igényel. Mivel a robot programja nyílt forráskódú, ezért azt bárki szabadon elérheti és módosíthatja saját igényei szerint.

1.5.2. Algoritmus implementálása

A kifejlesztett algoritmus implementálását a gyári program módosításával oldottam meg. Robot által rögzített kép felbontását 640×480-ra állítottam (ez a felbontás elegendő az algoritmusom biztonságos működéséhez), mivel a kamera 15-20 fps-el képes ilyen minőség mellett rögzíteni. Első feladat mindenekelőtt a robot által rögzített YUV422 formátumú kép szürkeárnyalatossá alakítása. Ezek után a korábban részletezett mintavételezési és egyéb algoritmusaimat kellett implementálni. Itt viszont figyelembe kellett venni néhány, a hardverhez köthető megszorítást. A roboton található 500 MHz-es központi egység és 32 MB SDRAM memória éppen elegendő az algoritmus végrehajtásához, azonban a Blackfin típusú mikrovezérlő alapértelmezetten nem támogatja a 64 bites számításokat, ezért a lebegőpontos számábrázolást a gyári programnak kell emulálnia. Ennek következményeként a lebegőpontos számokkal való műveletek igen lassúak, ezért az FFT-nek egy 32 bites fix számábrázolással való megvalósítását kellett kidolgozni.

1.5.3. Működés a roboton

A helyes működés elérésének érdekében számos kísérletet végeztem, különböző mintázatokat és eseteket vizsgáltam meg. 1.6. ábrán egy megfigyelő robot 3 másik társáról (*robot1*, *robot2*, *robot3* jelölésekkel) rögzített kép látható. Minden robot egyedi mintázatot hordoz, amivel már az azonosítás is végrehajtható. A rögzített kép feldolgozását maga a robot végzi és megjelöli a társakat a rögzített kép felső egyharmadán fehér színnel. Később a

módosított képet továbbküldi egy külső egységnek, ahol láthatóvá válik a társfelismerés eredménye és az esetleges hibák.

1.6. ábrán jól látható a takarás problémájának esete, amikor két (vagy több) robot olyan pozícióban áll a megfigyelő szemszögéből, hogy a két (vagy több) mintázatot egynek vesszük a társfelismerés után. Erre ad megoldást a robotok azonosítása (1.7. ábra). A korábban részletezett azonosítási eljárás eredménye jól látható az 1.7. ábrán, ahol a mintázat azonosítója meg lett jelölve a képen fehér színnel. Továbbá megtalálható az ábrán a detektált mintázatok magassága, a robotok becsült és tényleges távolsága a megfigyelőtől.

Az általam elvégzett kísérletek alapján kijelenthető, hogy a kifejlesztett társfelismerő és azonosító eljárás megbízhatóan működik a teszteléshez használt Surveyor SRV-1 robotokon, hozzávetőlegesen 4-5 fps-el.

1.6. ábra. Társfelismerési algoritmus futása SRV-1 roboton. A felismert robotok a kép felső egyharmadán fehér színnel vannak jelölve.

1.7. ábra. Társak azonosítása a módosított mintázatok segítségével. Az ábrán jelöltem a megfigyelő robottól valót tényleges és becsült távolságokat.

2. fejezet

Gyülekezési algoritmus globális információk nélkül

2.1. Irodalmi áttekintés

Az elmúlt néhány évtizedben a robotika egyik kedvelt kutatási területévé vált az autonóm elosztott rendszerű robot swarm feladatok kutatása. A swarm intelligencia alapfeladatokra adott, az irodalomban található jelenlegi megoldások általában olyan elméleti eredmények, amelyek a gyakorlatban nem, vagy erős korlátozások mellett alkalmazhatóak. A legintenzívebben kutatott feladatok a következők: robotok egyetlen pontba gyülekezése (vagy a lehető legkisebb területre) [28, 29, 30, 31, 32, 33, 34, 35, 36, 37], területbejáró algoritmusok alkalmazása egy bázispontból indulva [38], és hasznos részecskék (pl.: táplálék) begyűjtése ([39, 40]).

Robot swarm alapalgoritmusokkal kapcsolatos kutatásaim során a gyülekezés problémáját vizsgáltam meg tüzetesebben. A gyülekezés definíciója pontszerű robotok esetén a következő: tetszőleges alakzatból kiindulva véges idő alatt egyetlen pontban kell a robotoknak találkozniuk. A környezetet ebben az esetben akadálymentesnek tekintjük, tehát a robotoknak csak a társak elhelyezkedésével és mozgásával kell foglalkozniuk. A felvetett problémára kézenfekvő megoldás lehet egy konvergencia alapú megközelí-

tés, ez esetben feltételezzük, hogy minden robot belátja a teljes környezetet és mindenki által ismert az egész swarm átmérője. Az algoritmus végrehajtása során az átmérőt minden lépésben csökkentenünk kell a gyülekezés bekövetkezéséig. A fentiekben felvázolt megközelítés alkalmazásakor globális érzékelőknek kell segíteniük a robotok tevékenységét.

Kutatásaim során olyan gyülekezési algoritmusokkal foglalkoztam, melyekben csak lokális információk alapján tudnak a robotok a következő lépésükről dönteni. Ebben az esetben a gyülekezés már nem tekinthető triviális feladatnak, mivel semmilyen globális információ nem áll rendelkezésre. A feladat megoldhatóságának szempontjából meg kell határozni azokat a feladat specifikációjához szükséges jellemzőket, amelyekkel még végrehajtható a gyülekezés. Ezek a feladat megoldása szempontjából fontos jellemzők a következők:

- rendelkezik vagy nem rendelkezik memóriával,
- szinkrón vagy aszinkrón működésű,
- globális navigációs képességgel rendelkezik (közös koordináta rendszerrel) vagy sem,
- limitált a látótávolsága vagy az egész környezetét belátja,
- tudnak-e kommunikálni a robotok egymással vagy sem,
- pontszerű vagy kiterjedéssel rendelkező robotreprezentációt használunk.

A legtöbb gyülekezési algoritmust feledékeny (memóriával nem rendelkező) robotokra fejlesztették ki globális navigáció használata nélkül. A feledékenység itt azt jelenti, hogy a robot nem tud visszalépni az előző pozíciójába, mivel csak az adott pillanat érzékelése alapján számolja a következő lépés célvektorát. Mivel az előző lépést nem tárolja el, így az nem befolyásolja a jövőbeni lépések számításában. Ez az egyik olyan alapvető feltétel, amitől a gyülekezés problémája nem egy triviális feladat. Tipikusan egy gyülekezési algoritmusban a következő lépések ismétlődnek, amennyiben az egyes lépések szinkrónizálva vannak a robotok között:

- *Look*: látható robotok relatív pozíciójának meghatározása
- *Calculate*: következő pozíció számolása a *Look* fázisban gyűjtött adatok alapján
- *Move*: mozgás végrehajtása a kiszámított pozíció felé

Szinkrón működésnél a robotok egyszerre hajtják végre a soron következő lépést egy külső vagy belső szinkrónjel hatására. Belső szinkrón esetén nem kell a robotok között a kommunikációt megvalósítani, csupán meg kell állapítani azokat a maximális időintervallumokat, melyek alatt a körbenézés, következő lépés számítása, valamint a leghosszabb lépés végrehajtható. Minden robot a megfelelő lépéshez tartozó maximális időtartamig várakozik, még akkor is, ha nem kell lépnie sehova. Egy aszinkrón modellben azonban a robotok különböző időpontokban indíthatják a lépéseiket attól függően, hogy mikor fejezték be előző lépésüket.

Tömegközéppont (COG – Center Of Gravity) számításon alapuló algoritmust Cohen és Peleg [30] használt először szinkrón működésű robot swarm esetén, amely tetszőleges alakzatból kiindulva sikeresen hajtja végre a gyülekezést. Idealizált megoldásukban egy igen erős feltételezéssel élnek: a robotok látótávolsága nem korlátozott, így minden egyed belátja a teljes környezetet.

Cieliebak és társai [29] COG alapú algoritmus helyett a látható robotokra legkisebb ráhúzható kör középpontja felé történő elmozdulást alkalmazták (SEC - Smallest Enclosing Circle), és ezzel az algoritmussal megoldást adtak pontszerű robotok aszinkrón gyülekezésére feledékeny robotokat feltételezve. Ebben az esetben sincs látótávolság korlátozás bevezetve.

A fentiekben említett algoritmusok megoldást kínálnak a gyülekezés problémájára pontszerű és nem korlátos látótávolságú robotok esetén. A valós feltételek között is alkalmazható modellek kialakítása irányában tett lépésként értékelhető az Ando és szerzőtársai [28] által pontszerű robotok esetére kifejlesztett szinkrón működésű algoritmus, ahol a robotok csak limitált látótávolsággal rendelkeznek. A limitált látótávolságot feltételező megoldásokban központi szerepet játszik az úgynevezett láthatósági gráf. A gráf csomópontjai jelentik a robotokat, az élek pedig a robotok közötti

kapcsolatot abban az esetben, ha azok látják egymást. Habár Ando és szerzőtársai is SEC algoritmust használtak, egyetlen globális középpont helyett minden egyed egy-egy lokális középpont felé mozdul el, mivel minden robot a közvetlen környezete alapján számolta a következő lépését. Emellett minden robotmozgás korlátozott, mivel a sikeres gyülekezéshez elengedhetetlen, hogy a láthatósági gráf ne szakadjon meg (azaz a gráf összefüggő maradjon) az algoritmus végrehajtása során. A lépések mértékének ellenőrzése nélkül a swarm nem egy ponthoz gyűlik össze, hanem annyi pontba, ahány részre az eredetileg összefüggő gráf szétesett. A lokális SEC algoritmusuk működését szinkrón esetre bizonyították be.

Később Flocchini [31] és Souissi [32] társaikkal együtt mutattak be egy aszinkrón megoldást limitált látótávolság mellett. Azonban itt feltételezték, hogy a robotok az irányultsággal is tisztában vannak (iránytűvel vannak felszerelve), így ők egyfajta globális navigációs rendszert használtak a feladat megoldásához.

A szakirodalomban az egyik legfrissebb szinkrón és limitált látást alkalmazó megoldás Degener nevéhez [35] fűződik, aki a COG és SEC helyett lokális konvex sokszög alapján határozza meg a következő lépést. Ebben a megoldásban nincs globális navigáció, viszont a kommunikáció engedélyezett, így a robotok meg tudják osztani egymással jövőbeni lépéseiket.

A gyakorlati megvalósítás érdekében tett következő lépés a pontszerű robotreprezentáció elhagyása, azaz a kiterjedéssel rendelkező robotreprezentáció használata. Ebben az esetben a robotokat zárt körlapként értelmezzük, mely egy középponttal és R_s sugárral rendelkezik. Azonban ennek a módosításnak komoly következményei vannak, a robotok nem képesek egyetlen pontba gyülekezni, így az eredeti gyülekezési definíciót meg kell változtatnunk. Másik probléma, hogy egy robot blokkolni tudja egy másik robot mozgását, valamint egy robot takarhatja több társát is a megfigyelő szemszögéből (mivel az egyedek nem átlátszóak) hiába helyezkednek el a láthatósági sugáron belül.

Ezen a ponton újra kell definiálni a gyülekezés fogalmát kiterjedéssel rendelkező robotok esetében, és meg kell ismerkednünk a kontakt gráf fogalmával. A kontakt gráf csomópontjai (hasonlóképpen a láthatósági gráfhoz) a robotok lesznek, él pedig akkor alakul ki két csomópont között, ha a

két robot olyan közel helyezkedik el egymáshoz, hogy érintik egymást vagy egy meghatározott távolságon belül helyezkednek el. Czyzowicz és társai [34] a kontakt gráf felhasználásával a következőképpen határozták meg a kiterjedéssel rendelkező robotok gyülekezését:

- a zárt alakzatok kontakt gráfja összefüggő, és
- mindegyik robot látja a többi robotot.

Hozzá kell tenni, hogy ők legfeljebb 4 robotra oldották meg a gyülekezést, több robot esetén a definíció második része már nehezen vagy egyáltalán nem teljesíthető.

Chaudhuri [36], később pedig Cord-Landwehr [37] mutattak be egy olyan algoritmust, mely tetszőleges alakzatból hozta létre a kontakt gráfot kiterjedéssel rendelkező robotokkal. A felhasznált modellben a robotok rendelkeztek globális navigációval és a környezetet is teljesen belátták, ami azt jelenti, hogy a többi robotot átlátszónak tekintették.

Kutatásaim során elsősorban olyan gyülekezési algoritmusokkal foglalkoztam, ami feledékeny, kiterjedéssel rendelkező, limitált látótávolságú robotokra lett kifejlesztve [41, 42, 43, 44, 45]. Ezen kívül nincs szükség a robotoknak globális navigációra és kommunikációra sem. A meglévő megoldások közül teszteltem a lokális SEC algoritmus módosítás nélküli és egy általam módosított változatát is. Valamint bemutatok egy saját fejlesztésű algoritmust, mely jobb megoldást ad az irodalomban fellelhető megoldásokhoz képest [41, 42] a fentebb említett feltételekkel. A következő fejezetekben részletesen bemutatom az Ando-féle [28] algoritmust és az új megoldást [41, 42], végül a két algoritmus összehasonlítását és a szimulációs tesztek eredményeit.

2.2. SEC alapú gyülekezési algoritmus

Ebben a fejezetben részletesen bemutatom Ando és társai [28] által publikált gyülekezési algoritmust, mivel a saját megoldásom működési feltételei és megszorításai nagyon hasonlóak a [28] cikkben találhatókhoz. Továbbá

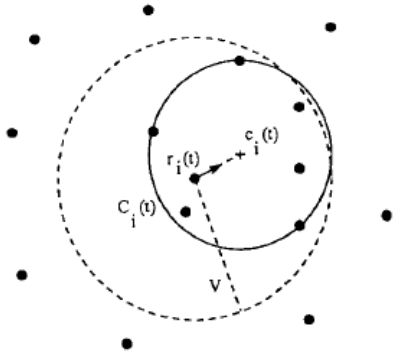
az általam készített szimulációban a saját algoritmusomat az Ando által közzétett megoldással és egy módosított változatával is összehasonlítottam.

A gyülekezési algoritmus a következő feltételek mellett működik: a robotok szinkrón működésűek, így minden egyed egyszerre indítja az egyes részfolyamatokat (környezet feltérképezése, következő pozíció számítása, mozgás a cél felé). A megszorítások közül az egyik legfontosabb, hogy a robotok nem rendelkeznek memóriával, mindig csak az aktuális érzékelés alapján kell dönteniük, nincs engedélyezve az előző pozícióba való visszalépés. Ezen kívül nincs lehetőség a robotok közötti kommunikációra és nem rendelkeznek globális navigációval sem. Továbbá fontos, hogy limitált a látótávolság, így mindenki csak egy V sugarú körön belül látja a szomszédjait. Megvalósítás szempontjából ők a pontszerű robotreprezentációt választották, ezért a gyülekezés leállási feltétele, hogy egyetlen pontban találkoznak a robotok. Ando és társai két részcélt definiáltak, amelyek a pontszerűségből fakadóan teljesíthetőek:

- minden robot egy összekapcsolt részgráfban (G_t) közelebb kerül egymáshoz $t + 1$-ben,
- azok a robotok, melyek t időpillanatban láthatóak egy megfigyelő szempontjából, $t+1$-ben is láthatónak kell lenniük, ahol t és $t+1$ két egymást követő ciklus indexei.

Az első részcél teljesítéséhez a látható robotokra legkisebb ráhúzható kör középpontja (SEC - Smallest Enclosing Circle) felé mozdulást alkalmazták, mely biztosítja, hogy minden lépésben egyre közelebb kerüljenek egymáshoz a robotok. 2.1. ábrán látható egy példa, ahol $r_i(t)$ a megfigyelő robot, V a láthatósági sugár, $C_i(t)$ a látható robotokra húzható legkisebb kör, $c_i(t)$ pedig ennek a középpontja.

Második részcél teljesítéséhez önmagában a SEC számítás nem elegendő, mivel ez a láthatósági gráf szétszakadásához vezetne. Így be kell vezetni a célvektor hosszának limitálását, ami megakadályozza, hogy különböző csoportokba gyülekezzenek a robotok.



2.1. ábra. $r_i(t)$ robot mozgása a legkisebb ráhúzható kör alapján. (Eredeti ábra [28])

2.2.1. Az összefüggőség biztosítása

SEC alapú gyülekezésnél, ha kizárólagosan csak a robot által számolt célvektort használjuk, az a swarm szétszakadását eredményezné, ezért szükségessé válik egy, a célvektort limitáló algoritmus bevezetése.

Jelölje $S_i(t)$ azon robotok halmazát, amik r_i robot által láthatóak t időpillanatban, valamint $r_i \in S_i(t)$. Legyen r_j $(i \neq j)$ egy robot $S_i(t)$–ből, amely r_i által látható t-ben. Legyen m_j a t időpillanatban az $r_i(t)$ és $r_j(t)$ robotok pozícióit összekötő szakasz felezőpontja. Ha a következő pozíciója r_i és r_j-nek a D_j zárt körlapba esik, ahol D_j középpontja m_j, sugara pedig $V/2$, akkor $t+1$-ben is látni fogja egymást az r_i és az r_j robot. Ezután r_i kiszámolja a maximális ℓ_j távolságot, amivel még nem hagyja el D_j-t, az irányultsága pedig marad a célvektor által meghatározott irány. Ha $dist(r_i(t), r_j(t)) = 0$ ($r_i(t)$ és $r_j(t)$ pontok távolsága t időpillanatban), akkor $\ell_j = V/2$ lesz, ellenkező esetben legyen $d_j = dist(r_i(t), r_j(t))$ a távolság és $\theta_j = \angle c_i(t) r_i(t) r_j(t)$ r_i a célvektor és a r_j közötti szög, ahol $0 \leq \theta_j \leq \pi$. Ebben az esetben a 2.1. képlettel számoljuk ℓ_j-t:

$$\ell_j = (d_j/2)\cos\theta_j + \sqrt{(V/2)^2 - ((d_j/2)\sin\theta_j)^2} \tag{2.1}$$

r_i minden $r_j \in S_i(t)$ robotra kiszámolja ℓ_j-ket és veszi a legkisebbet (2.2. képlet), valamint az $r_i(t)$ és $c_i(t)$ közötti távolságot a 2.3. képlettel.

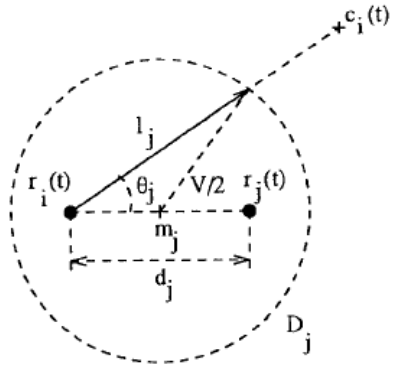
$$LIMIT = min_{r_j \in S_i(t) \setminus \{r_i\}} \{\ell_j\} \tag{2.2}$$

$$GOAL = dist(r_i(t), c_i(t)) \tag{2.3}$$

Végül a 2.4. képlettel meghatározzuk a lépés nagyságát.

$$MOVE = \min\{GOAL, LIMIT\} \tag{2.4}$$

A limitálás hatására az r_i robot D_j-n belül marad, és mivel minden robot ugyanazt az ágensprogramot futtatja, a láthatósági gráf nem fog megszakadni. 2.2. ábra demonstrálja a robotok elhelyezkedését és a célvektor limitálását.



2.2. ábra. Maximális távolság ℓ_j, amelyet r_i haladhat $c_i(t)$ irányába úgy, hogy D_j-t nem hagyja el. (Eredeti ábra [28])

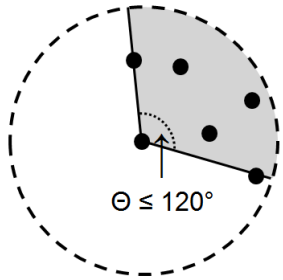
2.3. Saját szinkrón működésű gyülekezési algoritmus

Saját gyülekezési algoritmusom szinkrón működést feltételez, ahol a robotok kiterjedéssel rendelkeznek és nem átlátszóak. Továbbá a gyakorlati megvalósíthatóság érdekében a robotoknak limitált a látótávolsága és nincsenek globális navigációs rendszerrel felszerelve. Ezen kívül nem rendelkeznek memóriával és az egyes egyedek nem azonosíthatóak. Célom, hogy a felsorolt minimális tudással is végrehajtható gyülekezési algoritmust készítsek. Természetesen feltételezem, hogy a láthatósági gráf összefüggő az algoritmus indításakor.

Legyen $R = \{r_1, \ldots, r_n\}$ a robotok halmaza és $r_i(t)$ az i-dik robot pozíciója t időpillanatban. Minden robotot egy zárt körlapként értelmezünk a síkon (a robotok nem átlátszóak), ahol R_s a robot sugara, V pedig a láthatósági sugár.

Jelentős probléma a kiterjedéssel rendelkező robotok esetében, hogy a robotok takarhatják és blokkolhatják egymást mozgás közben. A kifejlesztett algoritmus alapötlete, hogy a robotok a látható legtávolabbi robot felé mozdulnak, amely egy eddig nem alkalmazott eljárás, használatával pedig részben megoldást kapunk a blokkolás problémájára is. A megvalósítás érdekében kettéosztottuk a robotok halmazát, ahol megkülönböztetünk periméter és belső robotokat. Periméter robotnak nevezünk minden olyan egyedet, mely legfeljebb 120°-os szög alatt lát szomszédos robotokat (tehát a swarm külső határán helyezkedik el) (2.3. ábra). Jelölje RP a periméter robotok halmazát, így megkaphatjuk a belső robotok halmazát ($R \setminus RP$) is, ezt jelölje RI. A robotok halmazát minden egyes ciklusban felbontjuk az előbb említett két részhalmazra.

Gyülekezési algoritmusom célja a következő: a periméter robotokat szeretnénk a lehető leggyorsabban a belső robotok felé mozdítani. Azonban önmagában az algoritmus működése azt eredményezné, hogy a láthatósági gráf felbomlik, és különböző csomópontok alakulnak ki. Viszont a belső robotok mozgásának lassításával és egy korlátozó algoritmus segítségével ez a probléma kiküszöbölhető.

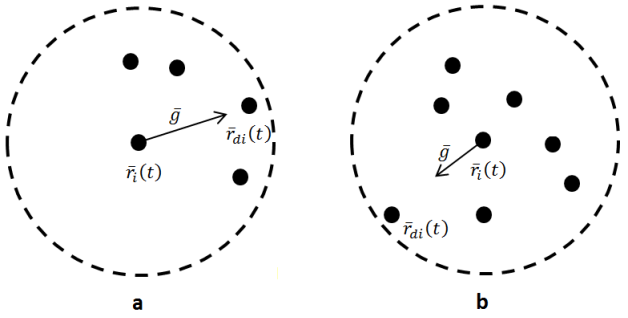


2.3. ábra. Periméter robot és az általa látott szomszédok.

Az algoritmus három fő feladatra bontható fel, amelyeket minden egyes lépésben végrehajtanak a robotok: *Look*, *Calculate*, *Move*. A *Look* fázisban a robotok összegyűjtik a látható szomszédjaikat t időpillanatban ($RV_i(t)$-vel jelöljük a későbbiekben), amit a $GetVisibleRobots$ eljárás hajt végre, ahol a bemenet R_t, a robotok halmaza t időpillanatban. *Calculate* fázisban meghatározza az aktuális robot magáról, hogy periméter vagy belső robot, ezután kiszámolja a legtávolabbi látható robot relatív pozícióját ($\bar{r}_{di}(t)$) a $GetFurthestVisbleRobot(RV_i(t))$ eljárás segítségével. A begyűjtött információk alapján mindegyik egyed meghatározza a célvektort ($\bar{g}$) a következő képlettel:

$$\bar{g} = c\frac{(\bar{r}_{di}(t) - \bar{r}_i(t))^2}{V} \tag{2.5}$$

ahol c 1-nek vagy 1/2-nek lett választva attól függően, hogy periméter vagy belső robotról van szó. A konstans értéke azért különbözik a két esetben, mert a peremen található egyedek körül kisebb sűrűségben helyezkednek el robotok és nagyobb sebességgel szeretnénk őket a többiek felé irányítani. Viszont a belső robotok esetében nincs szükség nagy lépésekre, mivel az ő feladatuk az, hogy bevárják a külső robotokat, és belül a lehető legegyenletesebben helyezkedjenek el.

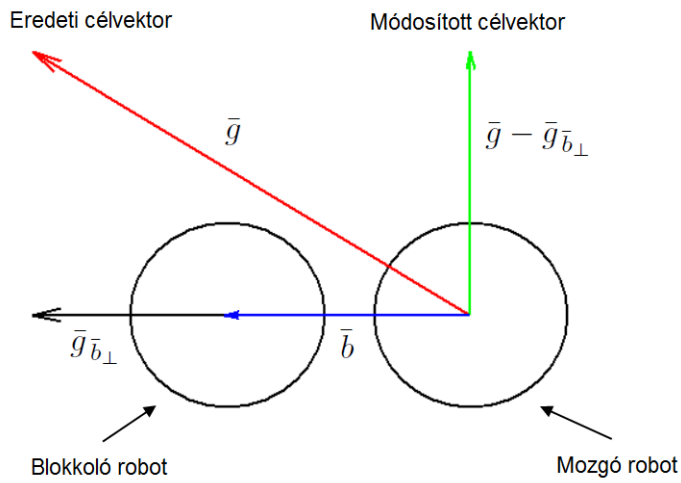


2.4. ábra. Periméter (a) és belső robot (b) lépései $c = 1$ és $c = 1/2$ mértékkel.

A láthatósági gráf megszakadásának elkerülése érdekében az Ando-féle [28] limitációs eljárást használtam, mely a szimulációs kísérleteink alatt megfelelően működött a fent részletezett algoritmus esetében is.

2.3.1. Blokkolási probléma feloldása

A kiterjedéssel rendelkező robot reprezentáció hátránya, hogy a robotok akadályoztatják egymást mozgás közben, így a robotok úgynevezett holtpont helyzetbe kerülnek. Ennek feloldása egy egyszerű módszerrel megvalósítható: a blokkolt robot eredeti célvektorát úgy módosítjuk, hogy az eredeti célvektor ($\bar{g}$) és a célvektor merőleges vetületének ($\bar{g}_{\bar{b}_\perp}$) különbségének irányába ($\bar{g} - \bar{g}_{\bar{b}_\perp}$) történik az új elmozdulás (2.5. ábra). Így a robot kellőképpen eltávolodik a blokkoló robottól és egyúttal közelebb kerül az eredeti céljához is. A módszert csak akkor alkalmazzuk, ha egy blokkoló robotunk van (a blokkoló robotokat a $GetBlockers(\bar{r}_i, RV_i)$ eljárás gyűjti össze) több blokkoló esetén nem, mivel lehetséges, hogy az aktuális robot már elérte a végső pozícióját. Természetesen amennyiben nincs blokkoló robot, akkor nem kell módosítani az eredeti célvektoron.



2.5. ábra. Megoldás a blokkolás problémájára egy blokkoló esetén.

2.3.2. Gyülekezési algoritmus

Ebben az alfejezetben megadom a gyülekezési algoritmusunk pszeudokódú leírását (5. algoritmus) szinkrón működésű robot swarm-okra. Az algoritmusban a *ComputeMovementLimitation* eljárás számolja az Ando-féle [28] mozgáslimitációt, a *TangentialDirection* pedig a kikerülést. Az aktuális robot mozgás vektorát $\bar{m}$-el fogjuk jelölni.

2.3.3. Tesztelés MATLAB szimulációval

Algoritmusom működését MATLAB szimulációval teszteltem, ahol három különböző eljárást (SEC, SEC kikerüléssel és az új algoritmus) teszteltem különböző kezdőállapotokból kiindulva, különböző robotszámokkal 12-től egészen 200 robotig.

Először a SEC algoritmussal valósítottam meg a gyülekezést, ami eredetileg pontszerű robotokra lett kifejlesztve. Ezután kiegészítettem a kike-

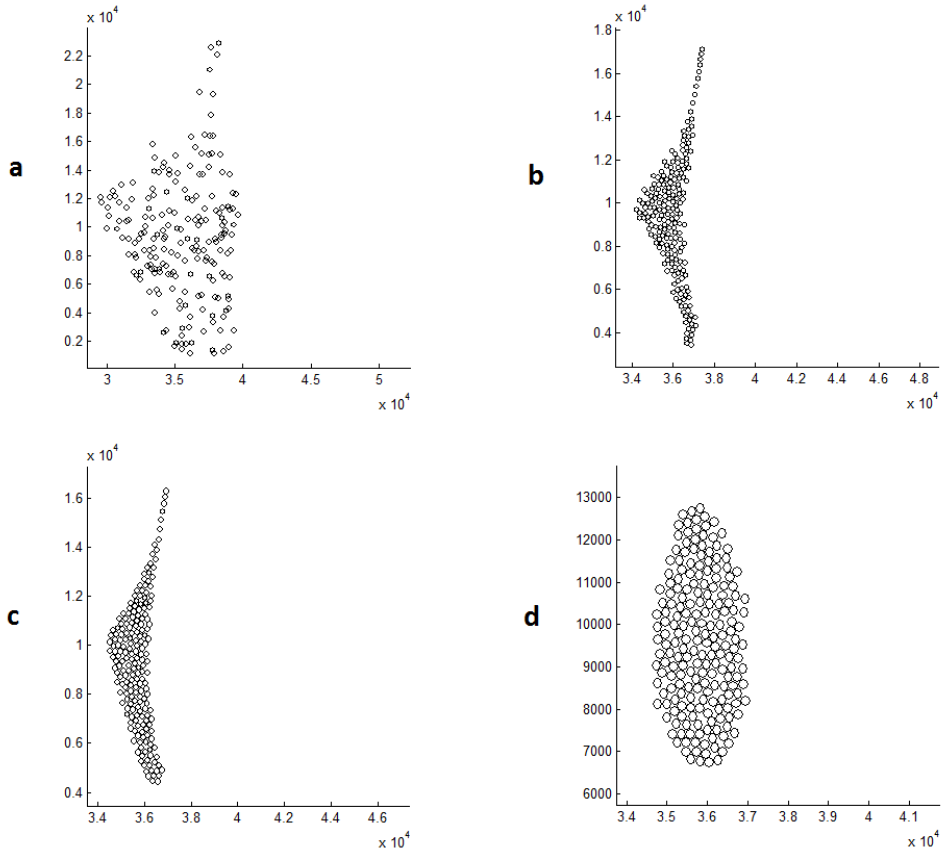
Algoritmus 5 Gyülekezési algoritmus ágensprogramja

Eljárás Gathering

1: $RV_i(t) = GetVisibleRobots(\bar{r}_i(t), R)$
2: $\bar{r}_{di}(t) = GetFurthestVisbleRobot(RV_i(t))$
3: **if** $\bar{r}_{di}(t) \in RP(t)$ **then** $c = 1$ **else** $c = \frac{1}{2}$
4: $\bar{g} = c\frac{(\bar{r}_{di}(t) - \bar{r}_i(t))^2}{V}$
5: $\bar{m} = ComputeMovementLimitation(\bar{r}_i(t), RV_i(t), \bar{g})$
6: $B_i(t) = GetBlockers(\bar{r}_i(t), RV_i(t)$
7: **if** $B_i(t)$ 1-nél több elemet tartalmaz **then** $\bar{m} = \bar{r}_i$
8: **else if** $B_i(t)$ 1 elemet tartalmaz
9: **then** $\bar{m} = TangentialDirection(\bar{r}_i(t), B_i(t), \bar{g})$
10: **else** r_i robot $\bar{m}$-el lép
11: **end if**
12: **Output:** r_i robot célvektora t időpillanatban

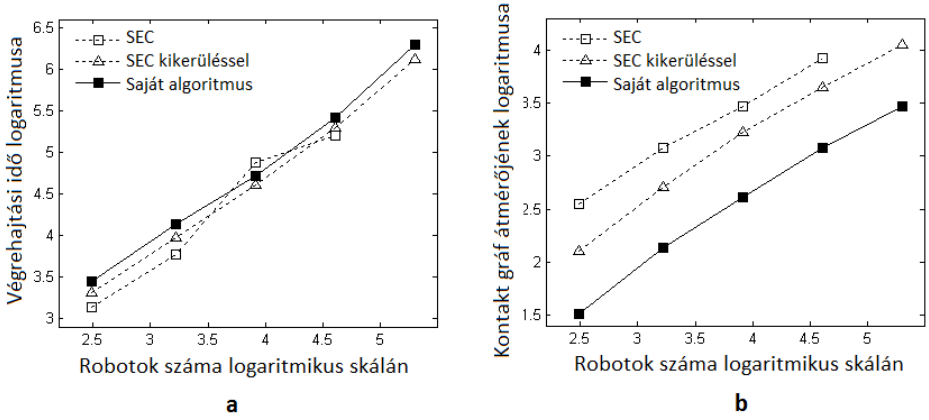
rülési eljárással annak érdekében, hogy a blokkolást elkerüljük, végül pedig a saját fejlesztésű algoritmust teszteltem. A három algoritmus $N = 12, 25, 50, 100, 200$ robotszámú swarm-on lett tesztelve, ahol minden swarmot 5 különböző kezdőpozícióból indítotottam. Kísérletek során vizsgáltam, hogy azonos kiindulások esetén milyen eredményeket kapok a három különböző gyülekezési megoldás alkalmazásával. Az algoritmusok leállási feltétele a következő volt: ha a swam-ban található összes robot kumulált lépéshossza elhanyagolhatóan kicsi, az algoritmus működését leállítjuk. A 2.6. ábrán látható az $N = 200$ eset egy lehetséges kiinduló pozíciója és a három gyülekezési algoritmus végeredménye, ahol jól látható a kontakt gráf kialakulása. A módosítás nélküli Ando-algoritmusnál [28] jól láthatóan a kezdeti állapotból egy elnyúlt alakzat jött létre (2.6/b. ábra). 2.6/c. esetben a blokkolás feloldási eljárással kiegészített Ando-algoritmussal már jobb eredményt értünk el, viszont az új algoritmus eredménye (2.6/d. ábra) adta a legtömörebb kontakt gráfot.

Végül a 2.7. ábrán láthatjuk a szimulációk kumulatív eredményeit, ahol mértem a gyülekezéshez szükséges időt (2.7/a) a kiinduló állapottól az al-



2.6. ábra. A gyülekezési algoritmusok eredménye $N = 200$ robotszám esetén. Az (a) mutatja a kezdőállapotot, a (b) az eredeti Ando-algoritmus, a (c) pedig az Ando-algoritmus végeredményét kiegészítve a kikerüléssel. Végül (d)-n láthatjuk a saját algoritmus futásának eredményét.

goritmus futásának a végéig, valamint a legnagyobb átmérőt (két legtávolabbi robot távolsága) a kialakult kontakt gráfban (2.7/b). Jól látható, hogy nincs szignifikáns különbség az egyes algoritmusok között a szükséges végrehajtási idő tekintetében, azonban a legnagyobb átmérő jóval kisebb az új algoritmus esetében. A változás jól látható a kikerüléssel módosított SEC algoritmusnál is, tömörebb alakzatot lehetett elérni.



2.7. ábra. Az átlagos, gyülekezéshez szükséges idők (a) és a legnagyobb átmérő alakulása a kontakt gráfban (b).

2.4. Aszinkrón és szinkrón algoritmusok összehasonlítása

2.4.1. Aszinkrón gyülekezési probléma

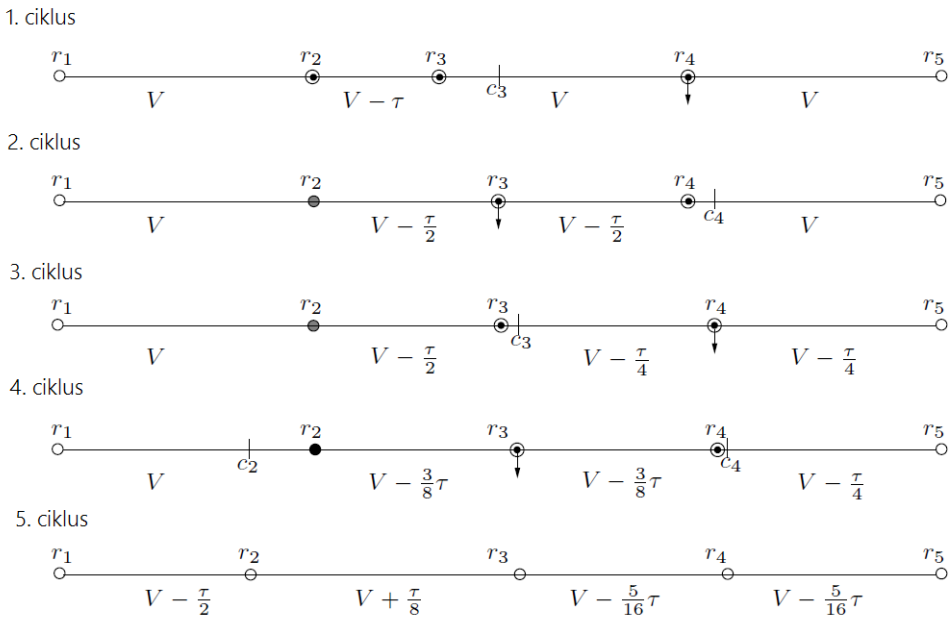
Aszinkrón algoritmus végrehajtásnál alapvetően megváltoznak a működési feltételek nagyszámú autonóm robot esetén, szemben a szinkrónnal, ahol az életciklus feladatok ($Look - Calculate - Move$) aktiválásai egyszerre történnek meg. Könnyen belátható, hogy a szinkrón algoritmusok a korábban

meghatározott minimális robottudás mellett nem fognak működni, mivel a gyülekezés sikeres végrehajtásához szükséges láthatósági gráf az algoritmus végrehajtása során, idővel meg fog szakadni.

A [46, 33] cikkek részletesen foglalkoznak az aszinkrón gyülekezés bonyolultságával és nem megvalósíthatóságával. Hozzá kell tenni, hogy a cikkekben található bizonyítások és kísérletek pontszerű robotreprezentációt feltételeznek, viszont [46] tanulmányban található egy olyan limitált látótávolságon alapuló kísérlet, amely kiinduló pontja saját kutatásaimnak a témában.

A feltételek azonosak voltak az Ando [28] cikkben leírtakkal, de a robot életciklusok különböző időpontokban indultak, tehát nem voltak szinkronizálva. Ahogy egy robot befejezi saját aktuális életciklus feladatát (elérte a kiszámított lokális célt), az életciklus újraindul a többi egyedtől függetlenül. Gyülekezési algoritmusnak szintén Ando megoldását használták fel.

A kísérlet során először is feltételezzük, hogy minden robot a minimális lépéshosszal tisztában van (δ), mindössze ennyi a robotok a priori tudása az algoritmus végrehajtása előtt. Itt az Ando által meghatározott két részcél közül az egyik (ha két robot $G(t)$ részgráfban t időpillanatban látja egymást, akkor $G(t+1)$-ben is láthatóak maradnak) megsértésére kapunk egy lehetséges esetet. Tételezzük fel, hogy 5 robotunk van, amelyek egy vonalban helyezkednek el (2.8. ábra). Továbbá vezessünk be egy konstanst δ néven úgy, hogy $\tau \leq \delta$ és $\delta = \tau/16$, ahol ha δ-nél kisebb a lépés, nem lépünk sehova, tehát a túl kicsi lépéseket nem engedélyezzük. Kezdetben a következő láthatóságot feltételezzük: r_1 és r_2 egymás láthatósági távolságán helyezkednek el ($dist(r_1, r_2) = V$), r_2 látja r_1-t és r_3-at ($dist(r_2, r_3) = V - \tau$), r_3 látja r_2-t és r_4-et ($dist(r_3, r_4) = V$), r_4 látja r_3-at és r_5-öt ($dist(r_4, r_5) = V$), végül r_5 látja r_4-et. A közvetkező pozíció számolásánál szintén Ando megoldását használták, tehát mindenki a kiszámított legkisebb, látható társak köré húzott körközéppont (c_i) felé halad, valamint a megszakadás elkerülése érdekében a lépéshossz limitálva van. A felvázolt kezdeti állapotból kiindulva az alábbi lépések hajtódnak végre:



2.8. ábra. Példa a láthatósági gráf megszakadására. A robotok egyenes vonalban helyezkednek el, ahol r_1 és r_5 robotok nem mozognak, r_2, r_3 és r_4 pedig Ando [28] algoritmusa alapján számol. Üres kör jelzi az inaktivitást (r_1 és r_5), a kör és benne pont a *Look* fázist, a telikör a *Calculate* fázist, a lefelé mutató nyíl pedig a mozgást jelöli az ábrán. Az r_i robot minden esetben a legkisebb befogadó kör középpontja (c_i) halad. (Eredeti ábra [46])

Ciklus 1. Összes robot, kivéve r_1 és r_5, belép az első *Look* fázisba, majd az érzékelés alapján a *Calculate*-be. A feltételezésünk az, hogy r_3 és r_4 gyorsabban számol, mint r_2. Az általuk kiszámolt célvektorok hossza:

$$r_3: \quad \begin{cases} Goal = dist(r_3, c_3) = |\frac{V-\tau+V}{2} - V + \tau| = \frac{\tau}{2} \\ Limit = \min\left\{-\frac{V-\tau}{2} + \frac{V}{2}, V\right\} = \frac{\tau}{2} \end{cases} \Rightarrow Move = \frac{\tau}{2}$$

$$r_4: \quad Goal = 0 \implies Move = 0$$

Tovább r_3 és r_4 elkezdenek a céljuk felé mozogni, miközben r_2 még mindig számol. r_1 és r_5 továbbra is $Wait$ állapotban marad.

Ciklus 2. Miután r_3 és r_4 mozgásának befejeztével a robotok ugyanúgy látják egymást, mint a kezdeti pozíciójukban. r_3 és r_4 újból végrehajtja a *Look* és *Calculate* fázisát:

$$r_3: \quad Goal = 0 \implies Move = 0$$

$$r_4: \quad \begin{cases} Goal = dist(r_4, c_4) = |\frac{V+V-\frac{\tau}{2}}{2} - V + \frac{\tau}{2}| = \frac{\tau}{4} \\ Limit = \min\left\{-\frac{V-\frac{\tau}{2}}{2} + \frac{V}{2}, V\right\} = \frac{\tau}{4} \end{cases} \Rightarrow Move = \frac{\tau}{4}$$

r_3 és r_4 újra elindul a kiszámított céljuk felé, amíg r_2 mindig az első *Calculate* fázisában van és r_1 és r_5 továbbra is $Wait$ állapotban van.

Ciklus 3. Előző ciklus végrehajtása után továbbra sem változtak a láthatóságok, tehát a láthatósági gráf továbbra is összefüggő. Ebben a ciklusban r_3 és r_4 belép a harmadik *Look* és *Calculate* fázisába.

$$r_3: \quad \begin{cases} Goal = dist(r_3, c_3) = |\frac{V-\frac{\tau}{2}+V-\frac{\tau}{4}}{2} - V + \frac{\tau}{2}| = \frac{\tau}{8} \\ Limit = \min\left\{-\frac{V-\frac{\tau}{2}}{2} + \frac{V}{2}, \frac{V-\frac{\tau}{4}}{2} + \frac{V}{2}\right\} = \frac{\tau}{4} \end{cases} \Rightarrow Move = \frac{\tau}{8}$$

$$r_4: \quad Goal = 0 \implies Move = 0$$

r_3 és r_4 elmozdul a célja felé, a többi robot pedig ugyanazt csinálja, mint az előző ciklusban.

Ciklus 4. Ebben a ciklusban is r_3 és r_4 végzi *Look* és *Calculate* feladatát.

$$r_3: \quad Goal = 0 \implies Move = 0$$

$$r_4: \quad \begin{cases} Goal = dist(r_4, c_4) = |\frac{V-\frac{3}{8}\tau+V-\frac{\tau}{4}}{2} - V + \frac{3}{8}\tau| = \frac{\tau}{16} \\ Limit = \min\left\{-\frac{V+\frac{3}{8}\tau}{2} + \frac{V}{2}, \frac{V-\frac{\tau}{4}}{2} + \frac{V}{2}\right\} = \frac{3}{16}\tau \end{cases} \Rightarrow Move = \frac{\tau}{16}$$

Továbbá r_3 és r_4 mozdul a célja felé. Eközben r_2 belép az első *Calculate* állapotába, viszont az értékek, ami alapján számolt még az első ciklusban található állapotokból jöttek.

$$r_2: \quad \begin{cases} Goal = dist(r_2, c_2) = |\frac{V+V-\tau}{2} - V| = \frac{\tau}{2} \\ Limit = \min\left\{V, -\frac{V-\tau}{2} + \frac{V}{2}\right\} = \frac{\tau}{2} \end{cases} \Rightarrow Move = \frac{\tau}{2}$$

Ezek alapján r_2 elkezdi *Move* fázisát és elmozdul a kiszámolt pozíció felé.

Ciklus 5. r_2 és r_3 közötti távolság erre a ciklusra $V + \tau/8 > V$ lett, így r_2 és r_3 nem látja többé egymást, a láthatósági gráf megszakadt. Ennek hatására megsérült az az előzetes kikötésünk, hogy aki t időpillanatban látható volt egy robot számára $t + 1$-ben is látható lesz, így a gyülekezés végrehajtása nem sikerült.

A kísérletből jól látható, hogy egyetlen robot elavult érzékelése és egyben lassabb számítása miatt a gyülekezés nem volt megvalósítható. Ez alapvető probléma dinamikusan változó környezetben működő autonóm robotoknál, ahogy láthattuk az érzékelés gyorsasága nagyban befolyásolja algoritmus végeredményét. Hozzá kell tenni, hogy az előbb részletezett gondolatkísérletet pontszerű robotokkal végezték, tehát az egyedek nem takarják egymást a megfigyelő szemszögéből. Viszont kiterjedéssel rendelkező robotoknál is ugyanezt az eredményt kapnánk, mivel mindenki csak a

mellette található társat érzékeli a limitált látótávolság miatt, és egy kivétellel mindenki nagyjából a látóhatósági sugár (V) távolságában helyezkedik el egymástól. Tehát, ha megváltoztatnánk a robotreprezentációt, viszont minden más feltételt megtartanánk, a lépések ugyanezt az eredményt hoznák és végül a láthatósági gráf kettészakadna az ötödik ciklusra.

2.4.2. Szinkrón gyülekezési algoritmusok alkalmazása aszinkrón esetben

Az előbbiekben láthattunk egy egyszerű példát, hogyan szakad meg a láthatósági gráf minimális robot tulajdonságokat feltételezve SEC algoritmus alkalmazásánál. Probléma alapvetően az érzékelés és ahhoz szükséges idő, valamint a limitált számítási teljesítményből adódik. Ezért feltételeztem, ha *Look* és *Calculate* végrehajtásához szükséges idők közelítenek a 0-hoz, akkor a megszakadások átlagos gyakorisága is csökkenni fog. Célom tehát az volt, hogy szimulációs kísérletekkel meghatározzam tapasztalati úton a megszakadás valószínűségét, meghatározott méretű swarm mellett. Ebben az esetben is minimális robot tulajdonságokat feltételeztem, valamint kiterjedéses robotreprezentációt használtam végig a kísérletek alatt.

2.4.3. Társak érzékelése

A társak érzékelésére az általam kidolgozott kamerakép feldolgozásán alapuló algoritmust használtam [47, 48, 49]. Pontosabban fogalmazva, a MATLAB szimulációk során feltételeztem, hogy a saját társfelismerő megoldásom segítségével határozzuk meg a lokális környezetben található egyedeket. Legtöbb algoritmus, amelyben limitált látótávolságot használnak, feltételezik hogy a robot 360°-ban látja be a környezetét. Mivel az általam javasolt megoldás a kamera képét dolgozza fel a társak felismerése és távolságok meghatározása érdekében, így azt kell feltételezem, hogy a robotokon található beépített kamera szenzor omnidirekcionális képet rögzít. Továbbá tételezzük fel a kép rektilinearitását is, ezért egy torzítatlan kép lesz a társfelismerési eljárás bemenete. Korábbi cikkeimben [47, 48, 49] VGA felbontással dolgoztam, amelyet egy Surveyor SRV-1 robot rögzített. A roboton

található szenzor 66.4 fokos látószöggel rendelkezik, ebből kiszámolható, hogy egy omnidirekcionális kép mérete azonos felbontással 3470×480 pixel lenne. Tehát feltételezzük, hogy a robot körbefordul saját tengelye körül, több képet is rögzít, amelyből végül elkészül a végső omnidirekcionális kép. Mivel a robot felismerő eljárásom 300 cm-ig képes megbízatóan működni, ezért a láthatósági távolságot $V = 300$ cm-re állítottam. A kísérlet során használt többi paramétert is a Surveyor robot technikai adatai alapján állítottam be: felhasználtam a robot sugarát ($R_s = 6$ cm), valamint a maximális utazási sebességét (40 cm/s).

Egy omnidirekcionális kép rögzítéséhez és feldolgozásához szükséges időt írjuk fel a következő alakban:

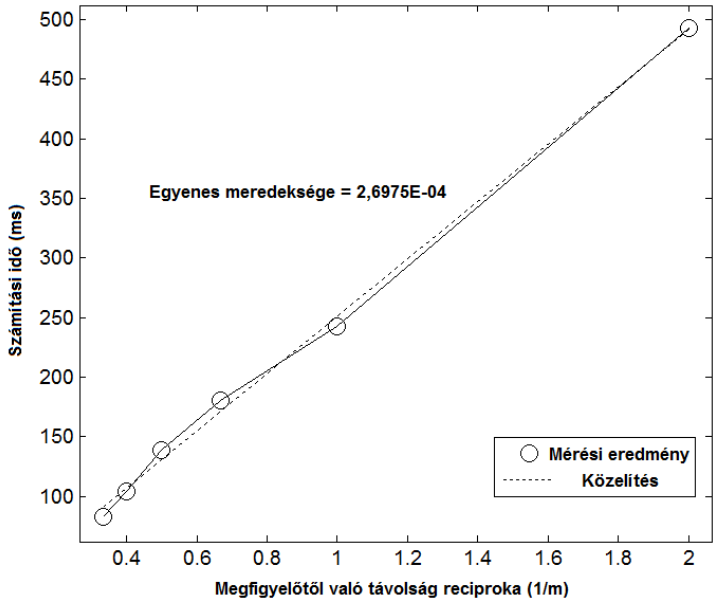
$$T = t_h + t_0 + \sum_{i=1}^{N} t_i(d_i). \tag{2.6}$$

ahol t_h a teljes képrögzítéshez szükséges időt jelenti (kamera szenzoronként változhat), t_0 pedig a teljes kép feldolgozásának ideje abban az esetben, ha egyetlen robotot sem lát a megfigyelő. Amennyiben 300 cm sugarú környezetben nincsenek társak a kép feldolgozási idejét megkapjuk $t_h + t_0$ –ból. Továbbiakban a számítási idő függ a látható társak számától és azok távolságától. A képletben N a látható robotok számát jelenti, az aktuális szomszéd pedig az i-dik robot. Amennyiben egy robot közel helyezkedik el a megfigyelőhöz, annál nagyobb mintázatot fog generálni a rögzített jeleneten minél közelebb van, szemben a távolabb található egyedekkel. Ebből következik, hogy ha az i-dik szomszéd távolsága (d_i) kicsi, akkor a feldolgozás idő ($t_i(d_i)$) megnő. Minél több szomszéddal rendelkezünk, annál többet kell számolni és a robot lassabban tud reagálni a környezet változásaira. Lineáris összefüggést vehetünk észre a távolság (d_i) reciproka és a mintázat alapján generált oszlopok száma között a képen:

$$t_i \sim 1/d_i. \tag{2.7}$$

Így a t_i-vel megadhatunk egy becslést egy társ felismerésének az idejére.

Számos kísérletet végeztem, amely során meghatároztam t_i és $1/d_i$ közötti összefüggést. A 2.9. ábrán láthatóak a kísérletek eredményei, melyeket

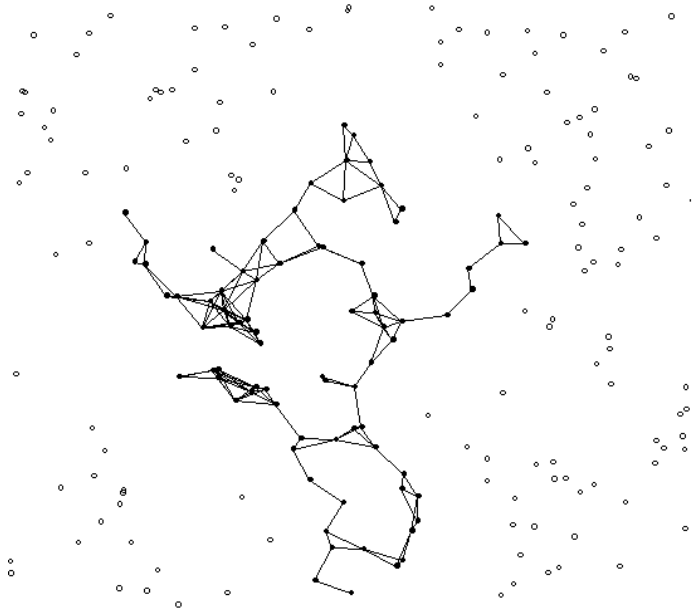


2.9. ábra. A megfigyelőtől való távolság és a számításhoz szükséges idő kapcsolata.

50 cm-től 300 cm-ig végeztem 50 cm-enként megismételve (az ábrán körrel jelöltem a kísérletek eredményeit). Szaggatott vonallal pedig a közelítő egyenes látható. Továbbiakban e közelítés alapján kapjuk a távolságokhoz tartozó számítási időket Surveyor SRV-1 robot esetén. A kísérletek során meghatároztam a kép rögzítésének idejét ($t_h = 305$ ms), valamint a minimális képfeldolgozási időt ($t_0 = 3339$ ms) is.

2.4.4. Kezdőpozíciók generálása

Korábbi cikktől eltérően ([41]) változtattam a robotok kezdőpontjainak generálásán. Feltételezzük, hogy a négyzet alakú környezetben generálunk egyenletes eloszlással robot pozíciókat egy előre meghatározott sűrűséggel ($1/9$ robot$/m^2$). Ezt a sűrűséget felhasználva a kapott átlagos robot-robot



2.10. ábra. Kezdeti robot pozíciók és a kiválasztott legnagyobb gráf.

távolságok közel lesznek a láthatósági sugár (V) mértékéhez. A generálás után kiválasztjuk a legnagyobb összefüggő láthatósági gráfot, amely a bemenete lesz a gyülekezési algoritmusnak. A 2.10. ábrán jól láthatók az egyenletes eloszlásból származó kezdőpozíciók, valamint a legnagyobb összefüggő gráf. Élek ebben az esetben is csak akkor jelennek meg a gráfban, ha a láthatósági sugáron belül helyezkednek a robotok.

Az eljárás egyenes következménye, hogy nem lehet előre megmondani pontosan a swarm tagjainak a számát (N). Erre a problémára egy lehetséges megoldás, ha különböző intervallumokat határoztam meg konkrét robotszámok helyett. Ezek a következők lettek: [5..9], [10..14], [15..19], [20..24], [25..29], [30..34], [35..39], [40..44], [45..49]. Minden esetben ugyanazt a sűrűséget használtam, intervallumonként pedig 100 láthatósági gráfot generáltam, így összesen 900 kezdőpozícióval dolgoztam.

2.4.5. Aszinkrón szimuláció eredményei

A szimulációk során Ando [28] szinkrón gyülekezési algoritmusát használtam aszinkrón feltételek mellett, mivel a [46] publikációban is erre az algoritmusra mutattak be egy speciális esetet a láthatósági gráf megszakadására. Összesen 3600 kísérletet végeztem MATLAB szimulációk segítségével, négy különböző robot számítási teljesítménnyel, a korábban generált 900 kezdőállapotból kiindulva. Célom az volt, hogy megvizsgáljam hogyan alakul a megszakadási valószínűség különböző számítási teljesítmények és robot számok mellett aszinkrón működésű swarm esetén.

T1-el jelöltem a Surveyor SRV-1 robot számítási képességét, tehát azt az időt, ami alatt a robot érzékeli a környezetét és meghatározza a társak helyzetét a korábban részletezett képfeldolgozási eljárás segítségével (T számolása a 2.6-os képlet alapján történik). További három nagyobb számítási teljesítményt is bevezettem, melyeket *T2*, *T3* és *T4*-el jelöltem. *T2* esetén 5-ször gyorsabb, *T3* és *T4* esetén pedig 10, illetve 20-szor nagyobb teljesítményt feltételeztem, amelyektől előzetesen azt vártam, hogy a megszakadások gyakorisága csökkenni fog és közelíteni fog a 0 megszakadási valószínűséghez. Fontos, hogy mindig a robot maximális utazási sebességét használtam minden egyes szimulációnál, viszont ugyanezeket az eredmények kaptuk volna 5-ször, 10-szer és 20-szor lassabb robotmozgások esetén is.

Összesen két leállási feltételt határoztam meg. Az első természeten az, amikor a swarm sikeresen összegyűlt, tehát a kontakt gráf létrejött. A második leállási feltétel az, hogy a láthatósági gráf megszakad a végrehajtás során. Minden egyes robot e két feltétel valamelyikének a bekövetkezéséig folytatja a $Look - Calculate - Move$ ciklus végrehajtását.

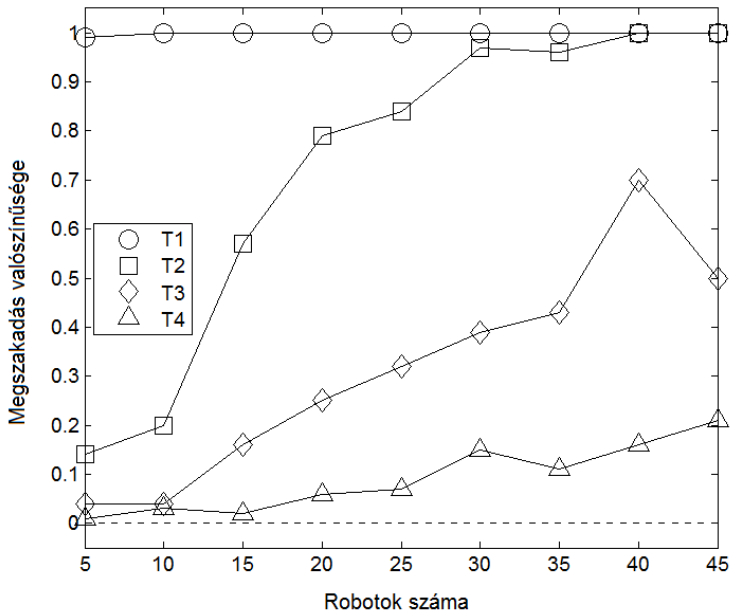
A kísérletek futási eredményeit megtekinthetjük a 2.1 táblázatban, valamint ugyanezek az adatok vizualizálva megjelennek a 2.11 ábrán is. Az eredményekből jól kivehető, hogy *T1* számítási teljesítmény mellett és a maximális utazási sebesség esetén a megszakadás valószínűsége nagyon közel van az egyhez, és egyre növelve a robotok létszámát ez az érték tovább romlik. Így kijelenthetjük, hogy *T1* számítási teljesítmény a megadott feltételek mellett nem alkalmazható. *T2*, *T3* és *T4*-nél viszont javulást ér-

zékelhetünk a teljesítmény növelésével, amely nem volt meglepő. *T4* gyorsaság esetén viszont jól látható, hogy $97 - 99\%$-os az összegyűlési arány 20 robotnál kisebb swarm-ok esetén, ami igen jó eredménynek számít. Ha a további eredményeket is megtekintjük, láthatjuk hogy a megszakadások száma együtt nő a swarm méretével, viszont a láthatósági gráf megszakadásának száma csökkenthető magasabb számítási kapacitás felhasználásával.

Az eredményekből arra a következtetésre juthatunk, hogy kiterjedéssel rendelkező robotok aszinkrón gyülekezése lehetséges, ha a ciklusokhoz szükséges végrehajtási időt csökkenteni tudjuk (*Look* és *Calculate* műveletek számítási ideje közelít a 0-hoz) és a megfelelő utazási sebességet használjuk. Ugyanilyen beállításokkal és azonos kiindulási pozícióból a pontszerű robotoknál a swarm nagyságrendekkel többször szétszakad, ami arra enged következtetni, hogy a kiterjedéses robotoknál az ütközés során tapasztalható ciklus szinkrónizáció (az ütköző robotok ciklusai újraindulnak) miatt a robotok nem tudják elhagyni egymás közvetlen környezetét. Természetesen ebben nagy szerep jut még a SEC alapú algoritmusnak is, ami a látható szomszédokra húzott legkisebb kör középpontja felé mozdítja a robotot.

	Swarm megszakadásának valószínűsége			
Intervallum	T1	T2	T3	T4
[5..9]	0.99	0.14	0.04	0.01
[10..14]	1.00	0.20	0.04	0.01
[15..19]	1.00	0.57	0.16	0.03
[20..24]	1.00	0.79	0.25	0.02
[25..29]	1.00	0.84	0.32	0.07
[30..34]	1.00	0.97	0.39	0.15
[35..39]	1.00	0.96	0.43	0.11
[40..44]	1.00	1.00	0.70	0.16
[45..49]	1.00	1.00	0.50	0.21

2.1. táblázat. Aszinkrón gyülekezés kísérleti eredményei.



2.11. ábra. Láthatósági gráf megszakadásának valószínűsége különböző számú robotok és számítási képességeknél. Szaggatott vonal jelöli a 0 megszakadási valószínűséget.

3. fejezet

Beltéri navigáció támogatása képi információk alapján

3.1. Bevezetés

Mobil robotokkal számos feladatot hajthatunk végre, ezek közül is a navigáció jelenti az egyik legnagyobb kihívást. A navigáció sikerességét alapvetően négy elem együttes működése határozza meg: érzékelés (a szenzorokból érkező adatokat értelmezi és kiválasztja azok közül a meghatározóakat), helymeghatározás (a robot meg tudja állapítani saját helyzetét a környezetében), felismerés (a robotnak tudnia kell hogyan érjen el egy megadott részcélt) és a mozgásvezérlés (a robotok motorjait úgy tudja irányítani, hogy az előzetesen számolt trajektóriát le tudja írni) [50]. A négy komponensből a helymeghatározás kapta a legnagyobb figyelmet az utóbbi évtizedben, mellyel kapcsolatban számos jelentős eredményt értek el a kutatók.

A lokalizáció (helymeghatározás) során a robot pozícióját meg szeretnénk becsülni annak munkakörnyezetében, ami azt jelenti, hogy a robot miközben felfedezi környezetét, egyúttal felépít egy térképet és meg tudja határozni a relatív elhelyezkedését a feltérképezett területen belül. Ezen kívül a kapott környezeti modell segíteni fogja a későbbi részcélok elérésében is. A helymeghatározás eredményét nagyban befolyásolja az érzékelők és

beavatkozók pontossága, amelyből a feladat nehézsége is adódik. A környezet érzékelésére többfajta szenzort is használhatunk, például GPS, vizuális érzékelőn alapuló mérést vagy ultrahangos érzékelőket. GPS-t elsősorban nagy területek feltérképezéséhez használhatjuk, mivel nagy hibával (akár több métert is hibázhat) adja meg az aktuális pozíciót. Így kisméretű vagy nanoméretes robotoknál nem használható, továbbá a technológiai korlátai miatt beltérben sem alkalmazhatjuk ezt a helymeghatározási formát. Beltéri navigáció esetén igen népszerű érzékelő az ultrahangos távolságérzékelő. Ennél az érzékelő típusnál is a mérés az érzékelő minőségétől függően valamilyen mértékű véletlen zajjal terhelt, ezért egyetlen mérés alapján nem lehet pontosan megállapítani az objektumok milyen messze találhatóak a megfigyelőtől. Véletlen zajból adódó hiba csökkentésére megoldás lehet, ha többször mérünk ugyanazzal a szenzorral, vagy ha több érzékelőt használunk és szenzorfúziót alkalmazunk.

Nem csak a szenzortechnológiából adódik a helymeghatározás nehézsége, ahogy a robot érzékelői zajos adatokat szolgáltatnak úgy a beavatkozók is zajjal terheltek. A robot már egy kicsi mozgás során növeli az elhelyezkedésének bizonytalanságát, viszont ha kellő figyelemmel van megtervezve a rendszerünk, akkor ezt a hatást minimálisra csökkenthetjük. A környezetben való navigálás pontosságán úgy tudunk javítani, ha az érzékelőktől kapott zajos adatokat megfelelően értelmezzük. Beavatkozók esetén odometriás eljárást (kerékelfordulás vizsgálata) vagy giroszkópot szoktak használni a halmozódó hibák csökkentésére. A beavatkozókból kapott hibák alapvetően a környezet modelljének hiányos ismeretéből adódnak, mint például a csúszós padló (a kerekek megcsúszhatnak), egy másik ágens (akár egy ember is) akadályozza a robot mozgását, stb. A hibákat kétfelé oszthatjuk, beszélhetünk determinisztikus (szisztematikus) és nem-determinisztikus hibákról. Szisztematikus hibáknak nevezzük azokat a hibákat, amelyek a rendszerünk modelljének hiányos ismeretéből fakadnak. Ezek a hibák rendszerint előre megjósolhatóak és hatásuk minimálisra csökkenthetőek. Ehhez a rendszerünket kalibrálni kell annak érdekében, hogy a hibákat okozó tényezőket megismerjük. Azonban számos nem-determinisztikus hiba (pl. véletlenszerű) fennmarad, ami továbbra is növeli a bizonytalanságot. Ezek a véletlenszerű hibák adódhatnak a rendszer kialakításából adódóan (pl.:

rossz színárnyalat érzékelése egy kamera esetében), rendszerint valószínűségi módszerek segítségével tudjuk modellezni őket.

Az odometriás eljárás pontosságának növelésére több megoldást is találhatunk. A szakirodalomban található eredmények közül három módszert emelnék ki. Borenstein és Feng [51], [52] UMBmark-nak nevezett eljárásában differenciál meghajtású mobil robotok odometriás képességeit mérték és javították annak pontosságát az érzékelők kalibrálásával. Megoldásuk lényege, hogy egy 4×4 méteres, négyzet alakú útvonalon mozognak a robotok először az óramutató járásával megegyező, majd ellenkező irányban. Minden 4×4-es útvonal végén megmérték a kezdő és végpozíció közötti távolság különbségét. Mivel az útvonal megtétele után a robotnak a kezdőpozícióban vagy annak közelében kell lennie, az eltérés alapján korrigálni lehet az odometriához tartozó paraméterek értékét. Ennek legfőbb előnye, hogy az eljárás egyszerű és nincs szükség további szenzorok használatára. Hátrányai közé sorolható, hogy a kezdő és végpozíció elméletben egy pozícióban van, valamint az útvonal megtétele során csak egyenes szakaszokon halad a robot és csak 90 fokos fordulatokat tesz.

Doh és társai [53] kidolgoztak egy új megoldást (PC-Method) az odometria okozta hibák korrigálására, mely alkalmas a determinisztikus (szisztematikus) és a nem-determinisztikus (nem szisztematikus) hibák csökkentésére egyaránt. Feltételezték, hogy a kalibrálni kívánt robot az odometriás eljáráson túl más érzékelővel is rendelkezik, amik segítik a lokalizáció megvalósításában. Az odometriás modell felépítésében nagy szerepet játszanak a további érzékelők, mivel az odometria és a többi érzékelő által kapott pozíció különbségekből számolni tudjuk az odometria hibáját. Az eljárás előnyei a korábbi eredményekkel szemben, hogy nagyon pontosan bekalibrálhatjuk az érzékelőinket és tetszőleges trajektóriát használhatunk a végrehajtáshoz. A több érzékelős megoldás egyúttal hátrányt is jelent, ezen kívül ugyanúgy, mint az UMBmark algoritmusnál itt is egy zárt trajektóriát kell használni a kalibrációhoz megegyező kezdő és végpontokkal.

Papadopoulos és Misailidis [54] munkájukban azt a cél tűzték ki, hogy egy új kalibrációs megoldást alkotnak, mely a korábbiakhoz képest rugalmasabban használható és egyúttal tovább növeli az odometria pontosságát. Az általuk kidolgozott eljárásban nincs szükség azonos kezdő és végpon-

tokra, és többfajta szenzor felhasználására sem. Egyetlen nagyon erős feltételük van, miszerint a robot kinematikai modellje előzetesen ismert és a hozzátartozó paramétereket is nagyon pontosan tudnunk kell. Amennyiben a paraméterek nem pontosan ismertek a végső pozíciót csak nagy hibával tudjuk megbecsülni.

Az odometriás eljáráson túl a helymeghatározás pontosságát tovább növelhetjük a környezetben elhelyezett jellemző objektumokkal (tereptárgyakkal) [50]. Jellemző objektumoknak olyan mesterséges jelöléseket nevezünk a környezetben, amelyeket a robot könnyen tud érzékelni. Ezek a passzív objektumok nagyon pontos lokalizációt tesznek lehetővé, segítségükkel a szenzorokból érkező hibák folyamatos összeadódására kapunk egy megoldást. Egyúttal azzal az erős feltételezéssel is élünk, hogy abban a pillanatban, amikor a robot eléri ezeket az objektumokat, a helymeghatározást alapvetően tökéletesnek tekintjük.

3.2. Javasolt navigációs megoldás

Az általam kidolgozott eljárásban a környezetben elhelyezett jellemző objektumok kamera képe alapján számoljuk a robot relatív pozícióját. A célom az volt, hogy olyan megoldással szolgáljak, amely egy jól bekalibrált odometriás eljáráshoz hasonló pontossággal működik, viszont annál általánosabban felhasználható legyen. Ez alatt azt értem, hogy ha több robotot használunk, az odometriás eljárásnál a robotokat egyesével be kell kalibrálni. Saját megoldásomban pedig a környezetre kell csak egyszer elvégezni a kalibrációt és azonos beállítások mellett akár különböző típusú robotokkal is használhatjuk.

A robot lokalizációjához és a terület feltérképezéséhez kizárólag a kamerából vett információkat kívánjuk használni, tehát a robotunk a vizuális érzékelő rendszerét fogja használni a helymeghatározáshoz. Feltételezzük, hogy a robot olyan beltéri környezetben dolgozik, amely sík padlóval rendelkezik, és a padló egy meghatározott mintázattal van borítva. Ez a mintázat a környezetben tevékenykedő robot számára előzetesen ismert. A robotra szerelt kamera fix pozícióban rögzít képeket a környezetről, ahol a kamerát

úgy helyezzük el, hogy számára a mintázat jól belátható legyen. A kamera segítségével a robot folyamatosan figyeli a környezetét, a rögzített képeken pedig képszegmentációval emeli ki a keresett objektumokhoz tartozó pixeleket. Mivel előzetesen már ismerjük a keresett mintázat tulajdonságait, így a pixelekből geometriai információkat tudunk kinyerni. A jellemző objektumok további jeleneteken való azonosításából és követéséből tudunk a későbbiekben információt kapni a robot elmozdulásáról, pozíciójáról, valamint a mintázat alapján a robot fel tudja térképezni környezetét is.

Természetesen az eljárás továbbfejleszthető több érzékelő felhasználásával. A vizuális érzékelő rendszert kiegészíthetjük az előbb említett odometriás eljárással, iránytűvel vagy más érzékelővel. Több érzékelő előnye, hogy minden egyes mérésnél több adat áll rendelkezésre, így robosztusabban és pontosabban működő helymeghatározást kapunk.

3.3. Kamera kalibráció és perspektív transzformáció

Algoritmusom használata előtt a roboton elhelyezett kamerát kalibrálni kell annak érdekében, hogy megismerjük a kamera belső paramétereit (fókusztávolság (f_x, f_y), a képsík és pixel koordináták közötti transzformáció és a főpont koordináták(p_x, p_y)), valamint a kamera lencséiből adódó geometriai torzítás jellemzőit. Jellemzően a lencse geometria kialakításából hordó vagy párna torzítást kaphatunk legtöbbször a keletkezett képen. Ebben az esetben különösen fontos, hogy rektilineáris (torzítatlan) képen dolgozhassanak a felhasznált képfeldolgozó eljárások.

A rektilineáris kép előállításához először definiálni kell a kamera belső paramétereit leíró mátrixot:

$$K = \begin{pmatrix} f_x & 0 & p_x \\ 0 & f_y & p_y \\ 0 & 0 & 1 \end{pmatrix} \tag{3.1}$$

Valamint a torzítási együtthatókat tartalmazó mátrixot:

$$D = \begin{pmatrix} k_1 & k_2 & k_3 & k_4 & k_5 \end{pmatrix} \tag{3.2}$$

Zhang kalibrációs megoldása [55] alapján a belső kamera paramétereit és a torzítási együtthatók értékeit meg tudjuk határozni egy kalibrációs tábláról (pl.: egy aszimetrikus sakktábla mintázat) készült X, Y és Z elforgatott képek segítségével. A kiszámolt paraméterek alapján már elő tudjuk állítani a torzítatlan képeket, amelyek alapvetően meghatározzák a további képfeldolgozási eljárások helyes működését.

Feltételezzük, hogy a robot egy akadálymentes környezetben dolgozik és a robot elejére fel lett erősítve egy kamera, amely a robot előtti területet képes érzékelni egy előre meghatározott szögben. Továbbá a teljes robot által bejárható környezet sík terep, tehát a robotnak csak egyenes padlós területen kell dolgoznia. Fixen beállított kamera dőlésszög és az előbbiekben felsorolt megszorítások mellett a kamera által érzékelt lokális környezetről tudjuk, hogy egy 2 dimenziós sík. Ebből következik, hogy lineáris transzformáció segítségével létrehozhatunk egy felülnézeti képet, amivel már pontosan lehet a későbbiekben dolgozni. A perspektív torzulás visszaszámlálásához egy 3×3-as transzformációs mátrix (H) szükséges, melyet az eredeti és a transzformált térből származó 4 pontpárból definiálunk. A transzformációs mátrixszal a következő összefüggést írhatjuk le az eredeti (x, y) és a transzformált kép pontjai (x', y') között, ahol a pontok homogén koordináta alakban vannak:

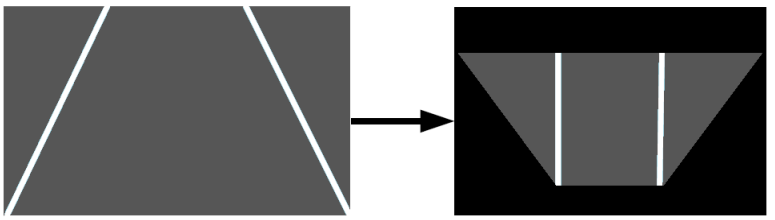
$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = H \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{3.3}$$

ahol H a homográfiát leíró mátrix:

$$H = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \tag{3.4}$$

H meghatározásához pedig a következő hibafüggvényt kell minimalizálni:

$$\sum_i \left(x'_i - \frac{h_{11}x_i + h_{12}y_i + h_{13}}{h_{31}x_i + h_{32}y_i + h_{33}} \right)^2 + \left(y'_i - \frac{h_{21}x_i + h_{22}y_i + h_{23}}{h_{31}x_i + h_{32}y_i + h_{33}} \right)^2 \quad (3.5)$$



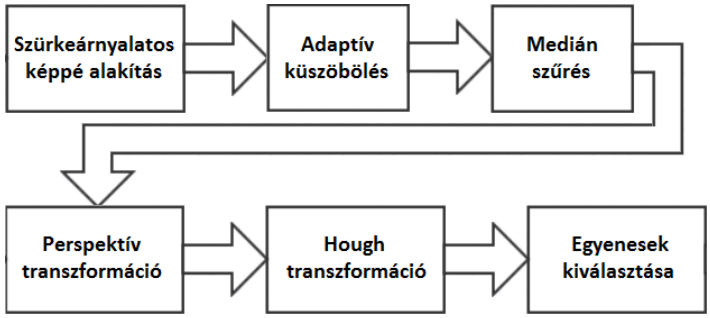
3.1. ábra. Perspektív transzformáció eredménye.

A transzformáció eredményét a 3.1. ábra mutatja, ahol a bal oldalon az eredeti, jobb oldalon pedig a torzítatlan képet láthatjuk. Ennek eredményeképpen olyan, a későbbiekben fontos információkat tudunk kinyerni a képből, mint a párhuzamosság vagy a merőlegesség, ahogy azt az ábrán is láthatjuk.

3.4. Jelenet előfeldolgozása

A jelenet feldolgozására több megoldás is született, viszont a disszertációmban csak a legutolsó, általam legjobbnak ítélt változat kerül részletezésre. Egy korábbi verzió már publikációban megjelent [56], főbb különbség a most bemutatott és a cikkben megjelent változat között a felhasznált képfeldolgozási módszerekben van.

A rögzített jelenet előfeldolgozásának lépései a 3.2. ábrán láthatók. Első lépésként mindhárom színcsatornát (RGB) felhasználva a kamera által rögzített színes képet szürkeárnyalatossá alakítom. Továbbiakban csak az így kapott intenzitásképen dolgozunk, ahol az egyes pixelek [0,255] intervallumon belül vehetnek fel értéket. Második lépésként az előre meghatározott



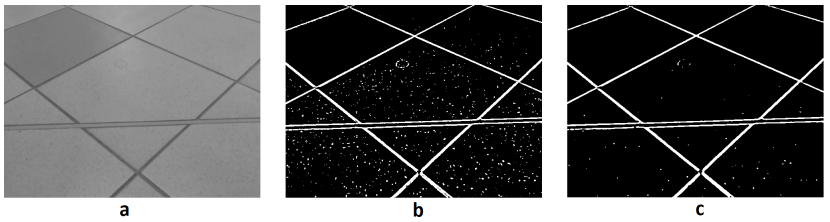
3.2. ábra. A robot által rögzített kép előfeldolgozásának lépései.

perspektív transzformációs mátrixszal kiszámoljuk a környezet felülnézeti képét. Feltételezzük, hogy a képen található, padlót borító objektumokat intenzitás alapon szegmentálni tudjuk, így elválasztva a hátteret az általunk keresett objektumok pixeleitől. Küszöbölés módszerével könnyen megvalósíthatjuk ezt az elválasztást, viszont a kísérletek eredménye alapján egyetlen globális küszöb használata az egész képre nem adott kielégítő eredményt. A háttér intenzitásértékei a rögzített kép több részén is különböző értéket vettek fel, ami a megvilágításból és a robot által vetett árnyékból adódott elsősorban. A problémára megoldást jelenthet, ha a küszöbölés folyamán nem a teljes képre, hanem annak csak egy részletére korlátozzuk a szegmentálást. Ezért a küszöbölés adaptív formáját hajtjuk végre a szürkeárnyalatos képen. Természetesen az adaptív küszöbölés megvalósítására több módszer is létezik, melyek eredménye nagyban különbözhet egymástól. Az általam használt adaptív küszöbölésnél a kiválasztott régión (ablakon) belül a következőképpen számoljuk a küszöbértéket (T): vesszük az ablakon belüli értékek keresztkorrelációját egy Gauss-szűrővel, majd a kapott értéket egy C konstansból kivonva kapjuk az ablakhoz tartozó T küszöböt. Jól látható, hogy három paraméter is szükséges: az ablak mérete (n), a Gauss-függvény szórása (σ) és a C konstans. Az i-dik képrészlethez tartozó küszöböt (T_i) a következő képlettel számolhatjuk:

$$T_i = \left[\sum_{\substack{(x',\, y') \in W_G \\ (x+x',\, y+y') \in F}} F(x + x',\, y + y') \cdot W_G(x',\, y') \right] - C, \tag{3.6}$$

ahol F a bementi intezítás kép, W_G pedig a Gauss-függvény diszkretizált alakja.

Az eredeti szürkeárnyalatos képet és az adaptív küszöbölés eredményét a 3.3. ábrán láthatjuk. Az adaptív küszöbölésből kapott képen találhatunk némi só és bors típusú zajt, melyet utófeldolgozással csökkenthetünk, illetve eltüntethetünk a képről. Erre a feladatra mediánszűrést használtam, mivel a legtöbb hiba eltüntetése mellett ez a módszer hagyta leginkább épen a mintázathoz tartozó egyeneseket alkotó pixeleket (3.3. ábra).

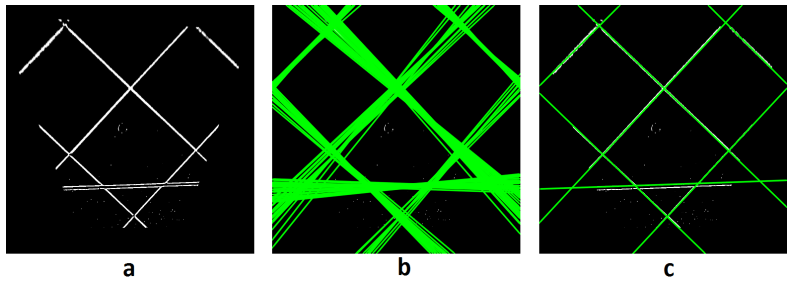


3.3. ábra. Kamera által rögzített kép szürkeárnyalatossá alakítva (a), majd az adaptív küszöbölés (b) és a medián szűrés eredménye (c).

Küszöbölés után a szűrt képen az előre kiszámolt transzformációs mátrix segítségével végrehajtjuk a perspektív transzformációt. Ennek eredménye, hogy a keresett struktúra párhuzamos és merőleges elemei a továbbiakban megfigyelhetővé válnak (3.4. ábra).

Az egyenesek érzékelésére a Hough-transzformációt [57] alkalmazzuk a felülnézeti képen, ahol az egyenesekről csak geometriai információkat tárolunk polárkoordináták formájában. r és θ paraméterekkel írunk le egy egyenest, ahol r sugár (pont origótól mért távolsága) θ pedig a szöge. A

Hough-transzformáció szintén használja a küszöbölést, amivel kiválaszthatjuk a pixelekre illeszkedő egyenesek közül a legjellemzőbbeket. Mivel egy ponthalmazra több egyenes is illeszkedhet, könnyen előfordulhat, hogy a kép egy részletére rengeteg egyenes jelöltet kapunk, melyek paraméterei csak kismértékben különböznek egymástól (3.4. ábra). Így egyetlen globális küszöb használata nem jelent megoldást, a küszöbölés után további teendőink lesznek. Hough-transzformációval nem csak egyszerű (egyenes, kör, ellipszis), hanem bonyolultabb alakzatokat [58] is tudunk keresni, mint például téglalap [59], parallelogramma [60] vagy parabola.



3.4. ábra. Felülnézeti kép (a), Hough-transzformáció által megtalált egyenesek (b) és a legjobban illeszkedők kiválasztása (c).

Az előfeldolgozás utolsó lépése szolgál a többszöri érzékelés hibájának javítására. Ebben a lépésben először a Hough által megtalált egyeneseket illeszkedés jósága alapján csökkenő sorrendbe rendezzük. Ezen a rendezett tömbön keressük meg a hasonló paraméterű egyeneseket és választjuk ki a legjobban illeszkedőt. Két egyenesről az alábbi módon dönthetjük el, hogy ugyanarra a pixelhalmazra illeszkednek:

$$(\Delta\theta < T_\theta \ \textit{és} \ \Delta r < T_r) \ \textit{vagy} \ (\Delta\theta > (2\pi - T_\theta) \ \textit{és} \ \Delta r < T_r) \qquad (3.7)$$

ahol T_r a sugárhoz tartozó, T_θ pedig a szöghöz tartozó küszöb. Továbbá $\Delta\theta$-t és Δr-t a következőképpen számoljuk:

$$\Delta\theta = |\theta_i - \theta_j| \tag{3.8}$$

$$\Delta r = |r_i - r_j| \tag{3.9}$$

Amennyiben a 3.7-es feltételünk igaz lesz, a két kiválasztott egyenes ugyanarra a pixelhalmazra fog esni.

Ahogy látható két küszöböt is használunk a kiválasztáshoz (T_r és T_θ), amelyek értékeit a mintázat ismeretében tudjuk meghatározni. Végeredményként a leginkább illeszkedő egyenes objektumokat kapjuk, ami jól látható a 3.4. ábrán. Továbbiakban csak a csökkentett számú (azaz minden pixel-egyenesre egyetlen illeszkedő) egyenesekkel dolgozunk, így a teljes mintázatfelismerő eljárás számítási ideje is kevesebb lesz.

3.5. A mintázat felismerése

A továbbiakban a padló rácsstruktúra tulajdonságait kihasználva fogunk dolgozni az előfeldolgozott képen. Az alapötletet Jung és Schramm munkája [59] adta. Ebben a cikkben téglalapok keresését ablakos Hough-transzformációval valósították meg. Egyenes primitívek és a Hough-paramétertér alapján elfordulástól és mérettől függetlenül téglalapokat tudunk vele keresni. Az eljárás mindössze két paramétert vár, a keresendő téglalapok minimális és maximális átmérőjét. A cikkben bemutatott eredményeknek csak egy részét tudtam alkalmazni, ezért egy új megoldást dolgoztam ki a teljes rácsstruktúra keresésére.

A Hough-transzformációból kapott egyenesek ($L(r, \theta)$) között az alábbi geometriai relációknak kell fennállni ahhoz, hogy egy rácselemet (négyzetet) beazonosíthassunk:

- Az egyenesek páronként jelennek meg, ahol az első pár L_1 és L_2 egyenesek párhuzamosak ($\theta_1 \approxeq \theta_2$), valamint a második L_3 és L_4 is párhuzamosak ($\theta_3 \approxeq \theta_4$).

- Ahol a szögek megegyeztek (tehát az egyenesek párhuzamosak) az egyeneseknek egymástól meghatározott távolságra kell lennie, így ki tudjuk szűrni azokat a párokat melyek túl közel vagy távol vannak egymástól: $|r_1 - r_2| > T_{rmin}$ és $|r_1 - r_2| < T_{rmax}$, ahol T_{rmin} és T_{rmax} küszöbökkel szabályozhatjuk a megengedett sugár eltérést.

- Két párhuzamos pár egyenesei egymással $\Delta\theta \approxeq \frac{\pi}{2}$ szöveg zárnak be.

Amennyiben a fenti feltételek teljesülnek, akkor találtunk egy rácselemet, ezért a következő lépéseket kell végrehajtanunk. Legyen az $L_1(r_1, \theta_1)$, $L_2(r_2, \theta_2)$ ... $L_n(r_n, \theta_n)$ egyenesek. Kiválasztunk két egyenest és vizsgáljuk azok párhuzamosságát:

$$\Delta\theta = |\theta_i - \theta_j| < T_\theta \tag{3.10}$$

$$\Delta r = |r_i - r_j| > T_{rmin} \tag{3.11}$$

$$\Delta r = |r_i - r_j| < T_{rmax} \tag{3.12}$$

Minden i, j párra a következőket hatjuk végre: először a θ paraméter alapján vizsgáljuk a párhuzamosságot, ahol ha T_θ küszöbnél kisebb a két szög különbségének abszolút értéke, akkor $\theta_i \approxeq \theta_j$ áll fenn. T_{rmin} küszöbbel kiszűrhetjük az egymáshoz nagyon közel elhelyezkedő egyeneseket, T_{rmax}-szal pedig a nagyon távol esőket. Utóbbi két küszöbnél feltételezzük, hogy a rácselemek méretei előre ismertek. E három feltétel teljesülése esetén a két kiválasztott egyenes párhuzamos lesz a keresett rácsstruktúrában. Miután az összes pár rendelkezésre áll, kiszámoljuk azok $P_k(\xi_k, \alpha_k)$ értékét, ahol

$$\alpha_k = \frac{1}{2}(\theta_i + \theta_j) \tag{3.13}$$

$$\xi_k = \frac{1}{2}|r_i + r_j| \tag{3.14}$$

Utolsó lépésként összehasonlítjuk az összes párt jellemző P értékeket és megvizsgáljuk, hogy a párhuzamos párok egymással merőleges szöget zárnak be. Így ha találtunk egy téglalapot teljesülni fog az alábbi feltétel:

$$\Delta\alpha = \|\alpha_l - \alpha_l| - \frac{\pi}{2}| < T_\alpha \tag{3.15}$$

ahol T_α-val állapítjuk meg P_k és P_l párról, hogy merőlegesek-e.

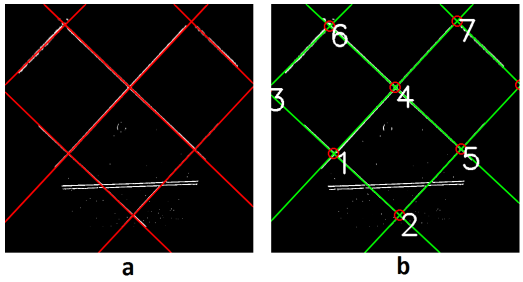
A mintázat felismerésének eredményét a 3.5. ábra mutatja. Jól látható, hogy a korábbi 3.4. ábrán található vízszintes, nem a mintázathoz tartozó egyenes objektumot már elhagytuk, így az előbb részletezett mintázat azonosító eljárás megfelelően működött az előfeldolgozásból kapott hibás adatokon is.

3.6. Jellemző objektumok követése

A jeleneten felismert objektumokat (pl.: egyenesek, egyenesek metszéspontjai) fel kell címkézni és tárolni abból a célból, hogy azonosítani és követni tudjuk őket a későbbi jeleneteken. A továbbiakban csak azokkal az objektumokkal szeretnénk dolgozni, melyek a padlót borító rácsokhoz tartoznak, a többit pedig nem vesszük figyelembe. Ezek az objektumok lesznek az egyenesek, valamint az egyenesek metszéspontjai.

Objektumok első felcímkézésekor az észlelés sorrendjétől függően minden objektum kap egy egyedi azonosítót. További érzékeléskor távolság metrikát (ebben az esetben az Euklideszi távolságot) felhasználva azonosítjuk be a jeleneteken a már mentett objektumainkat, számoljuk az aktuális és az előző érzékelés objektumai közötti távolságokat. Amennyiben a távolság kisebb, mint egy előre definiált konstans, az azonosítást sikeresnek tekintjük. Új objektumot találtunk abban az esetben az aktuális jeleneten, ha az előző érzékelés egyik objektumával sem azonosítható be. Természetesen az érzékelés hibájából adódóan előfordulhat, hogy egy-két jelenet erejéig eltűnnek az objektumok. Ezeket nem töröljük azonnal, hanem számoljuk, hogy az egymást követő jeleneteken hányszor nem tudtuk senkivel sem megfeleltetni. Ha egy előre meghatározott értéken túl nem érzékeljük az adott elemet, töröljük a felcímkézett objektumok közül.

Az egyenesek azonosítása után számoljuk azok metszéspontjait és hasonlóképpen címkézzük és követjük ezeket az objektumokat is.



3.5. ábra. Rácshoz tartozó egyenesek kiválasztása (a), a képen található jellemző pontok megjelölése és felcímkézése (b).

3.7. Trajektória felépítése

Robotunk, a feladata végrehajtása közben bizonyos időközönként eltárolja a két érzékelés közötti elmozdulás vektorát annak érdekében, hogy felépítse a megtett utat reprezentáló trajektóriát. Tekintettel arra, hogy a robot folyamatosan feldolgozza a kamera érzékelőjéből származó jeleneteket, a jellemző objektumokat felcímkézi és követi azokat (az algoritmus végrehajtása során minden egyes felcímkézett metszéspontnak eltárolom a korábbi pozícióját, annak érdekében, hogy megállapítsam a robot elmozdulását), a trajektória felépítése triviális feladat. Megoldásomban a felcímkézett jellemző pontokat csökkenő sorrendre rendezzük az alapján, hogy hányszor érzékeltük őket a korábbi képeken. Első mintavételezésnél minden pontot felhasználva számoljuk az elmozdulást (a pontok jelenlegi és előző mintavételezés során felvett képkoordinátája alapján), későbbiekben pedig csak azokat vesszük figyelembe, amik egy előre meghatározott küszöbnél többször megtaláltunk már a képen. Az általam végzett kísérletek esetében ez azt jelentette, hogy legalább 50 jeleneten keresztül érzékeltük és azonosí-

tottuk azokat az objektumokat, melyek alapján számoljuk az elmozdulás vektorát. Minden mintavételezésnél a számolt vektorok átlagát tároljuk, így végül a mentett vektorokból fog állni a trajektória, ami a robot által megtett utat írja le.

3.8. Kísérletek és eredmények

3.8.1. Kamera kalibrálása

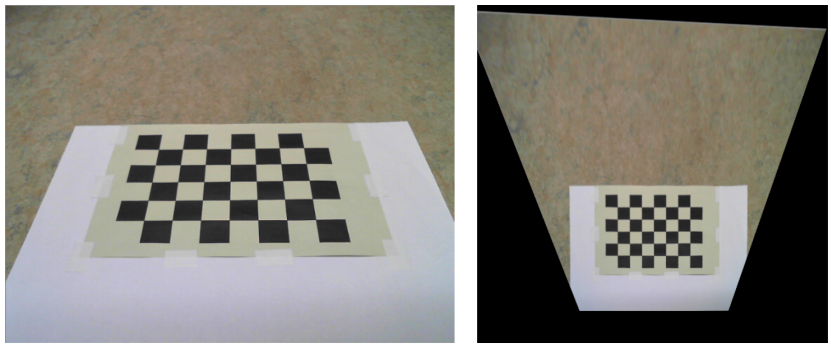
A kísérletek során 1080p HD-érzékelővel felszerelt MS LifeCam Studio webkamerával dolgoztam. Mivel a kamera automatikus élességállítással (autófókusszal) rendelkezik, így azt szoftveresen ki kellett kapcsolni. Először beállítottam a kísérletek során használt dőlésszögre a kamerát és az autófókusz segítségével az éles képet, majd elmentettem a kapott kamera fókuszt és manuálisra állítottam az élességállítást. Erre azért volt szükség, mert a kamera kalibráció (és az algoritmus futása) során feltételezzük, hogy az eszközünk fix fókuszbeállítással rendelkezik, ellenkező esetben minden fókuszhoz végre kellene hajtani a kalibrációs eljárást. Az eszköz által biztosított felbontások közül a VGA (640×480) felbontást tartottam megfelelőnek mind a kép minőségét illetően, mind pedig a szükséges számítási teljesítmény szempontjából.

A kamera kalibrálása során 7×5-ös aszimmetrikus sakktáblamintát használtam. Magát a kalibrációt OpenCV [61] segítségével végeztem előre elkészített képek (40 db) segítségével, ahol X, Y és Z irányban elforgatott kalibrációs tábláról készített képek voltak a bemenetek.

3.8.2. Perspektív transzformáció

Perspektív transzformáció mátrix számolásához ismét az OpenCV-t és a kalibrációs táblát használtam. A kamerát beállítottam arra a szögre, melyet a kísérletek folyamán is használtam (30°), majd a kalibrációs táblát úgy helyeztem el, hogy jól belátható legyen a kamera számára. Az OpenCV függvények közül a *warpPerspective* eljárást használtam, amelynek elegendő 4 pontot megadni a bementi képről és ezek megfelelőjét transzfor-

málté képen. A bementi képről a kalibrációs tábla négy legszélső sarokpontját választottam, melyeket pixel alatti pontossággal határoztam meg a $findChessboardCorners$ szintén OpenCV eljárással. Végül meg kellett adni a bementi sarokpontok transzformált megfelelőit is, itt felhasználtam a minta fizikai méreteit és úgy alakítottam át, hogy a végeredmény elférjen az 500×500 pixelt tartalmazó képen. A transzformáció eredménye a 3.6. ábrán látható.



3.6. ábra. Perspektivikus transzformáció számítása a kalibráláshoz is használt sakktábla mintázattal.

3.8.3. Az algoritmus működésének tesztelése

A kifejlesztett algoritmus helyes működésének ellenőrzéséhez ismét kísérletekre volt szükség. Elsődleges célom a javasolt eljárás pontosságának ellenőrzése volt, illetve annak vizsgálata, hogy a hibák milyen mértékben adódnak össze nagyobb távolság megtétele után, ezáltal mennyire befolyásolják a végeredményt. Saját megoldásom pontosságát a [52], [54], [62] cikkekben leírt kalibrált odometriás eljárással hasonlítottam össze, ahol a relatív hiba 2.0E-3 volt. A relatív hibát $\Delta L/L$ képlettel definiáljuk, ahol ΔL jelenti a robot pozíciójának hibaátlagát, miután a robot megtett egy L

hosszúságú utat [63]. Odometria esetén ez a relatív hiba konstans lesz, tehát a megtett úttal arányosan a pozíció becslésében a bizonytalanság nőni fog.

A kísérletek beállításánál azt feltételeztem, hogy a robot egyenes vonalú egyenletes mozgást végez 20 m hosszúságú szakaszon. 2 és fél méterenként rögzítettem, hogy melyik kameraframe-nél tart a felvétel, így a teljes 20 m-es szakaszon belül 8 mérföldkövet határoztam meg, ez a későbbiekben fontos lesz a pontosság méréséhez. Összesen 15 tesztszakasz került felvételre, melyeket utólag dolgoztam fel az erre a célra kifejlesztett feldolgozó programommal. A kísérletek közül 10 kísérlet szolgált az algoritmusom kalibrálására, a fennmaradó 5 pedig a pontosság ellenőrzésére.

Tesztelés során a korábban már említett problémák (robot által vetett árnyék, különböző fényviszonyok, kisebb hibák a padlón) nem befolyásolták döntően a megtett távolság becslését, az algoritmusom ezekben az esetekben is robosztus működést mutatott. A kalibrálásra használt adatsort feldolgozva pixelben megkaptam a mérföldkövekhez tartozó távolságokat, ebből már számolhatók voltak a cm/pixel arányok mérföldkövenként. A mért adatok átlaga adta azt az arányt, amivel a későbbiekben minden pozícióban becsültem a megtett távolságot. Előzetesen azt feltételeztem, hogy a kialakított eljárás olyan pontos, hogy determinisztikus hiba nem befolyásolja a kapott eredményeket (tehát a rendszer megfelelően lett kalibrálva). A 3.1. táblázatban foglaltam össze tételesen a kalibrációhoz használt adatsorból kapott eredményeket: átlagosan becsült távolságot, a szórást és a relatív hibát.

A szórási adatokon jól látható, hogy a kezdőpozíciótól való távolság növekedésével a bizonytalanság mértéke (habár kis mértékben) nőtt. Megtett távolsághoz viszonyítva viszont a relatív hiba csökkent, így a javasolt eljárás az odometriával ellentétben nem lesz konstans bármekkora megtett távolság esetében.

A fennmaradó 5 tesztadatsoron is a kalibrálás során kapott cm/pixel aránnyal dolgoztam, az eredményeket a 3.2. táblázatban összegeztem. Ebben az esetben a relatív hiba tovább csökkent, ez betudható annak is, hogy a kalibrálás adatok felvételekor a robot mozgása nem volt teljesen pontos, így a mért adatok tartalmazhattak valamekkora bizonytalanságot. A tesztelő

adatok eredményei jól mutatják, hogy a determinisztikus (szisztematikus) hibát szinte minimálisra sikerült csökkenteni, mivel 20 m megtétele után csak 3,39 cm a bizonytalanság. Így kijelenthető, hogy a rendszert megfelelően sikerült beállítani, és sikeresen elértem egy tökéletesen bekalibrált odometria 2.0E-3 relatív pontosságát ([52], [54], [62]).

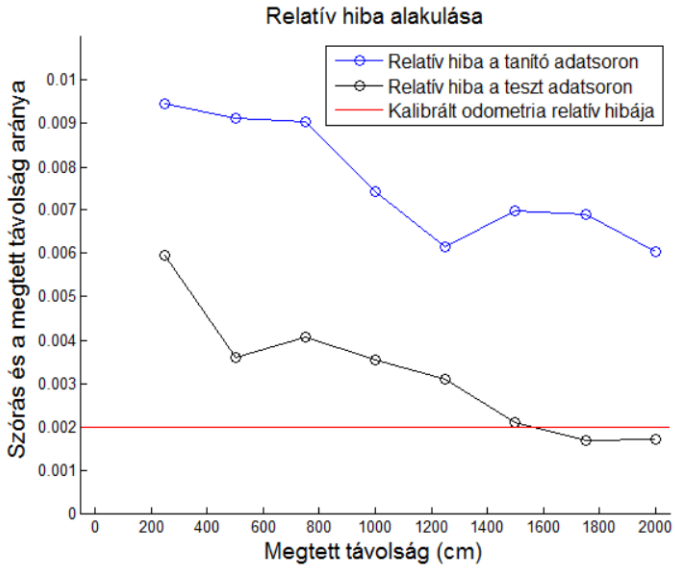
	Mért távolság		
Megtett távolság	Átlag	Szórás	Relatív hiba
250 cm	250.1927 cm	2.3678 cm	0.0095
500 cm	503.2030 cm	4.5560 cm	0.0091
750 cm	751.6248 cm	6.7867 cm	0.0090
1000 cm	1000.1458 cm	7.4261 cm	0.0074
1250 cm	1244.8821 cm	7.6894 cm	0.0062
1500 cm	1495.6075 cm	10.4735 cm	0.0070
1750 cm	1745.8446 cm	12.0604 cm	0.0069
2000 cm	1993.8697 cm	12.0895 cm	0.0060

3.1. táblázat. A tanító adatsoron mért távolságok átlaga, szórása és a relatív hiba az egyes mérföldköveknél.

	Mért távolság		
Megtett távolság	Átlag	Szórás	Relatív hiba
250 cm	249.9410 cm	1.4884 cm	0.0059
500 cm	500.4110 cm	1.7941 cm	0.0035
750 cm	750.4917 cm	3.0614 cm	0.0040
1000 cm	1001.4019 cm	3.5532 cm	0.0035
1250 cm	1247.5371 cm	3.8871 cm	0.0031
1500 cm	1497.2962 cm	3.1376 cm	0.0020
1750 cm	1744.9016 cm	2.9602 cm	0.0016
2000 cm	1995.0885 cm	3.3998 cm	0.0016

3.2. táblázat. A teszt adatsoron mért távolságok átlaga, szórása és a relatív hiba az egyes mérföldköveknél.

Relatív hibák alakulását összegeztem a 3.7. ábrán, ahol külön jelöltem a kalibrálás és a tesztelés eredményét és ezeken belül a mérföldkövekhez tartozó relatív értékeket is. Összehasonlításképpen a kalibrált odometria konstans relatív hibáját (2.0E-3) egy piros színű egyenessel jelöltem. A tesztelési eredményeken jól látható, hogy a relatív hiba kisebb megtett távolság esetén nagyobb, mint az általunk elvárt, viszont a megtett út növekedésével a kalibrált odometria pontosságát először megközelíti, majd az alá megy.



3.7. ábra. Relatív hibák a kalibrációhoz és teszteléshez használt adatsorokon.

3.8.4. A hiányos adatsor problémája

A felmerülő problémák közül még egyet érdemes megemlíteni, ami nagyban befolyásolhatja a kimenetként kapott trajektóriát. Előfordulhat, hogy az előfeldolgozás eredménye nem ad értékelhető eredményt, így a további lépéseket sem lehet végrehajtani. Ennek eredménye, hogy az érzékelési hibákból adódóan hiányos lesz az adatsorunk és a becsült megtett távolság további hibákkal terhelődik. Ilyenkor a mintavételezési fázisban nincsen elegendő számú adat, mely alapján számolni tudnánk a jellemző pontok elmozdulását, a feldolgozás pedig újraindul minden ilyen esetben. Amennyiben két mintavételezés között volt mért adat és előfeldolgozási hiba, csak részben tudjuk becsülni a megtett távolságot a jellemző pontok alapján. Ezért ezeket az adatokat sem érdemes elmenteni. Erre a problémára több megoldást is adhatunk, lehet szó akár utófeldolgozásról vagy már a feldolgozás közbeni adat kiegészítésről. Előbbi esetben a teljes mért adatsor rendelkezésünkre áll, a hibás részeket interpolációval meghatározhatjuk. Utóbbinál pedig, ha mérési hibát érzékelünk, becsülni tudjuk a múltbéli adatok (robot sebességvektora) alapján az aktuálisan megtett távolságot. A tesztelések során viszont arra jutottam, hogy egyszer sem kellett alkalmazni az adatsort javító eljárást abban az esetben, ha a paramétereket megfelelően állítottam be. Ez azt jelenti, hogy a kísérletek során nem tapasztaltam előfeldolgozásból adódó hibát, viszont a robusztusabb működés érdekében érdemes beépíteni az előbb említett eljárást.

Összegzés

Disszertációmban ismertettem azokat a főbb eredményeket, melyeket doktori tanulmányaim alatt értem el, kutatási témától függően három fejezetre bontva. Minden fejezetet szakirodalmi áttekintéssel kezdtem, ismertettem az általam relevánsnak ítélt publikációkat és azokban leírt eredményeket. A fejezetek további tartalma minden esetben a saját kutatási munkám ismertetése és az elért eredmények részletezése volt.

A 2. fejezetében mobil robot swarm-ok társfelismerési és azonosítási eljárásait vizsgáltam és tettem ajánlást egy új algoritmusra. Kutatásaim során sikerült egy olyan megoldást találni a társfelismerés problémára, amelylyel a swarm-ban található robotokat tudtuk érzékelni és azok távolságát megbecsülni. A módszer minimális módosításával lehetséges volt a robotok azonosítása is. Az eljárás alapja, hogy minden robotot egy ismétlődő mintázattal láttunk el, valamint feltételeztük, hogy a robotok kamerával rendelkeznek. A rögzített jeleneteket képfeldolgozási algoritmusokkal dolgoztam fel ügyelve arra, hogy maga a robot hardvere is képes legyen azokat feldolgozni külső számítógép használata nélkül. Az algoritmus helyességének ellenőrzését Surveyor SRV-1 robotokkal végeztem, a javasolt eljárást a robotra implementáltam.

A 3. fejezetben a swarm intelligencia alapfeladatait, azon belül is a gyülekezés problémáját részleteztem. Saját eredményeim közé tartozik egy olyan szinkrón gyülekezési algoritmus, kiterjedéssel rendelkező robotokra, amely sikeresen teljesíti a gyülekezést. A saját és az irodalomban található megoldásokat összehasonlítottam egy MATLAB-ban írt szimulációs

programmal. Az összehasonlítást két szempont alapján végeztem: gyülekezéshez szükséges idő és végső alakzat tömörsége.

Egy másik munkámban kiterjedéssel rendelkező robotok aszinkrón gyülekezési viselkedését vizsgáltam szintén szimuláció segítségével. A cél az volt, hogy megvizsgáljam, hogy a robotok számítási teljesítményének változása hogyan befolyásolja a gyülekezés sikerességét nem pontszerű robotreprezentációnál. Továbbá arra a kérdésre kerestem a választ, ha bizonyos lépésekhez szükséges számítási kapacitást elhanyagolhatónak veszünk, mekkora valószínűséggel teljesül a gyülekezés.

Utolsó fejezetem a mobil robotok beltéri navigációjával és lokalizációjával foglalkozott. Saját területfeltérképező eljárásomban kamera érzékelőt használtam és feltételeztem, hogy az egyenes padlós környezet, ahol a robot végrehajtja feladatát, egy meghatározott mintázattal van borítva. A mintázat alapján megbecsülhető, hogy mekkora volt a robot elmozdulása, valamint a kiinduló állapothoz képest az elfordulása. A jelenetek feldolgozásához képfeldolgozási eljárásokat alkalmaztam, figyelembe véve azt, hogy a környezet mintázata előzetesen ismert. Algoritmusom működésének helyességét kísérletekkel ellenőriztem és a kapott eredményeket összehasonlítottam a kalibrált odometriás eljárás pontosságával.

Irodalomjegyzék

[1] I. Rekleitis, G. Dudek, and E. Milios. Multi-robot collaboration for robust exploration. *Annals of Math. and Artifical Intelligence*, 31:7–40, 2001.

[2] D. Wilking and T. Röfer. Realtime object recognition using decision tree learning. *Lecture Notes in Computer Science*, 3276:556–563, 2005.

[3] U. Kaufmann, G. Mayer, G. Kraetzschmar, and G. Palm. Visual robot detection in robocup using neural networks. *Lecture Notes in Computer Science*, 3276:262–273, 2005.

[4] Y. Han and H. Hahn. Visual tracking of a moving target using active contour based ssd algorithm. *Robotics and Autonomous Systems*, 53:265–281, 2005.

[5] T. Laue and T. Röfer. Integrating simple unreliable perceptions for accurate robot modeling in the four-legged league. *Lecture Notes in Computer Science*, 4434:474–482, 2007.

[6] I. Engedy and G. Horváth. A global camera–based mobile robot localization. *Proc. of the 10th International Symposium of Hungarian Researchers on CINTI*, pages 217–228, 2009.

[7] A. Fabisch, T. Laue, and T. Röfer. Robot recognition and modeling in the robocup standard platform league. *Proc. of the Fourth Workshop on Humanoid Soccer Robots*, 2010.

[8] R.M. Haralick, K. Shanmugam, and I. Dinstein. Textural features for image classification. *IEEE Transactions on Systems, Man, and Cybernetics SMC-3*, 6:610–621, 1973.

[9] S. Arivazhagan and L. Ganesan. Texture classification using wavelet transform. *Pattern Recognition Letters*, 24(9-10):1513–1521, 2003.

[10] B.S. Manjunath and W.Y. Ma. Texture features for browsing and retrieval of image data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(8):837–842, 1996.

[11] G.M. Haley and B.S. Manjunath. Rotation-invariant texture classification using a complete space-frequency mode. *IEEE Transactions on Image Processing*, 8(2):255–269, 1999.

[12] D.A. Clausi and M.E. Jernigan. Designing gabor filters for optimal texture separability. *Pattern Recognition*, 33(11):1835–1849, 2000.

[13] F. Bianconi and A. Fernandez. Evaluation of the effects of gabor filter parameters on texture classification. *Pattern Recognition*, 40(12):3325–3335, 2007.

[14] T. Matsuyama, S.-I. Miura, and M. Nagao. Structural analysis of natural textures by fourier transformation. *Computer Vision, Graphics, and Image Processing*, 24(3):347–362, 1983.

[15] A.A. Ursani, K. Kpalma, and J. Ronsin. Texture features based on fourier transform and gabor filters: an empirical comparison. *Proc. of International Conference on Machine Vision*, page 67–72, 2007.

[16] T. Leung and J. Malik. Representing and recognizing the visual appearance of materials using three-dimensional textons. *International Journal of Computer Vision*, 43(1):29–44, 2001.

[17] M. Varma and A. Zisserman. A statistical approach to texture classification from single images. *International Journal of Computer Vision*, 62(1-2):61–81, 2005.

[18] J. Shotton, J. Winn, C. Rother, and A. Criminisi. Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context. *International Journal of Computer Vision*, 81(1):2–23, 2009.

[19] A. Speis and G. Healey. An analytical and experimental study of the performance of markov random fields applied to textured images using small samples. *IEEE Transactions on Image Processing*, 5(3):447–458, 1996.

[20] H. Deng and D.A. Clausi. Gaussian mrf rotation-invariant features for image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(7):951–955, 2004.

[21] T. Ojala, M. Pietikainen, and T. Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):971–987, 2002.

[22] M. Li and R.C. Staunton. Optimum gabor filter design and local binary patterns for texture segmentation. *Pattern Recognition Letters*, 29(5):664–672, 2008.

[23] Z. Guo, L. Zhang, and D. Zhang. Rotation invariant texture classification using lbp variance (lbpv) with global matching. *Pattern Recognition*, 43(3):706–719, 2010.

[24] M. Donoser and H. Bischof. Using covariance matrices for unsupervised texture segmentation. *Proceedings of 19th IAPR International Conference on Pattern Recognition*, 2008.

[25] I. Z. Yalniz and S. Aksoy. Unsupervised detection and localization of structural textures using projection profiles. *Pattern Recognition Letters*, 43(10):3324–3337, 2010.

[26] K. Bolla, T. Kovács, and G. Fazekas. A barcode based kin recognition and identification method for robot swarms. *Scientific Bulletin of*

"Politehnica" University of Timişoara, BS-UPT TACCS, 56(70):11–20, 2011.

[27] V. K. Ingle and J. G. Proakis. *Digital Signal Processing using MATLAB.* Thomson Bookware Companion Series, 2007.

[28] H. Ando, I. Suzuki, and M. Yamashita. Formation and agreement problems for synchronous mobile robots with limited visibility. *International Symposium on Intelligent Control*, page 453–460, 1995.

[29] M. Cieliebak, P. Flocchin, G. Prencipe, and N. Santoro. Solving the robots gathering problem. *LNCS, Springer, Heidelberg*, 2719:1181–1196, 2003.

[30] R. Cohen and D. Peleg. Robot convergence via center-of-gravity algorithms. *LNCS, Springer, Heidelberg*, 3104:79–88, 2004.

[31] P. Flocchini, G. Prencipe, N. Santoro, and P. Widmayer. Gathering of asynchronous robots with limited visibility. *Theoretical Computer Science*, 337:147–168, 2005.

[32] S. Souissi, X. Defago, and M. Yamashita. Using eventually consistent compasses to gather oblivious mobile robots with limited visibility. *LNCS, Springer, Heidelberg*, 4280:484–500, 2006.

[33] G. Prencipe. Impossibility of gathering by a set of autonomous mobile robots. *Theoretical Computer Science*, 384:222–231, 2007.

[34] G.J. Czyzowicz, L. Gasieniec, and A. Pelc. Gathering few fat mobile robots in the plane. *Theoretical Computer Science*, 410:481–499, 2009.

[35] B. Degener, B. Kempkes, and F.M. Heide. A local $o(n^2)$ gathering algorithm. *Proceedings of the 22nd ACM Symposium on Parallelism in Algorithms and Architectures*, page 224–232, 2010.

[36] S. Chaudhuri and K. Mukhopadhyaya. Gathering asynchronous transparent fat robots. *LNCS, Springer, Heidelberg*, 5966:170–175, 2010.

[37] A. Cord-Landwehr, B. Degener, M. Fischer, M. Hullmann, B. Kempkes, A. Klaas, P. Kling, S. Kurras, M. Martens, F. M. Heide, C. Raupach, K. Swierkot, D. Warner, C. Weddemann, and D. Wonisch. Collisionless gathering of robots with an extent. *LNCS, Springer, Heidelberg*, 6543:178–189, 2011.

[38] R. Cohen and D. Peleg. Local spread algorithms for autonomous robot systems. *Theoretical Computer Science*, 399:71–82, 2008.

[39] P. Valdastri, P. Corradi, A. Menciassi, T. Schmickl, K. Crailsheim, J. Seyfried, and P. Dario. Local spread algorithms for autonomous robot systems. communication and swarm intelligence issues in a swarm microrobotic platform. *Robotics and Autonomous Systems*, 54:789–804, 2006.

[40] S. Nouyan, A. Campo, and M. Dorigo. Gathering path formation in a robot swarm self-organized strategies to find your way home. *Swarm Intelligence*, 2(1):1–23, 2008.

[41] K. Bolla, T. Kovács, and G. Fazekas. Gathering of fat robots with limited visibility and without global navigation. *Swarm and Evolutionary Computation, LNCS, Springer*, 7269:30–38, 2012.

[42] K. Bolla, T. Kovács, and G. Fazekas. Kiterjedéssel rendelkező autonóm robotok gyülekezése. *Proc. of AGTEDU 2012*, 2012.

[43] K. Bolla, T. Kovács, and G. Fazekas. Autonóm robotok gyülekezése lokális tömegközéppont alapú algoritmus használatával. *Gradus (Journal of Kecskemet College)*, 1, 2014.

[44] K. Bolla, T. Kovács, and G. Fazekas. Local center of gravity based gathering algorithm for fat robots. *Issues and Challenges of Intelligent Systems and Computational Intelligence, Springer*, pages 175–183, 2014.

[45] K. Bolla, T. Kovács, and G. Fazekas. Investigating the rate of failure of asynchronous gathering in a swarm of fat robots with limited visibility. *Lecture Notes in Artificial Intelligence, Springer*, 8468:249–256, 2014.

[46] G. Prencipe. Instantaneous actions vs. full asynchronicity: Controlling and coordinating a set of autonomous mobile robots. *ICTCS 2001*, 2202:154–171, 2001.

[47] K. Bolla, T. Kovács, and G. Fazekas. Compact image processing-based kin recognition, distance measurement and identification method in a robot swarm. *Proc. of ICCC-CONTI 2010 IEEE Conference*, pages 419 – 424, 2010.

[48] K. Bolla, Z. Istenes, T. Kovács, and G. Fazekas. A fast image processing based robot identification method for surveyor srv-1 robots. *Proc. of 2011 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM2011)*, page 1003 – 1009, 2011.

[49] K. Bolla, T. Kovács, and G. Fazekas. A fast image processing based kin recognition method in a robot swarm. *1st Scientific and Expert Conference TEAM*, pages 165–170, 2010.

[50] R. Siegwart and I. Nourbaksh. Introduction to autonomous mobile robots. page 181–253, 2004.

[51] J. Borenstein and L. Feng. Correction of systematic odometry errors in mobile robots. *Proceedings of International Conference on Intelligent Robots and Systems*, page 569–574, 1995.

[52] J. Borenstein and L. Feng. Measurement and correction of systematic odometry errors in mobile robots. *IEEE Transactions on Robotics and Automation*, 12:869–880, 1996.

[53] N. Doh, H. Choset, and W. K. Chung. Accurate relative localization using odometry. *Proceedings of International Conference on Robotics and Automation*, 2:1606–1612, 2003.

[54] E. Papadopoulos and M. Misailidis. On differential drive robot odometry with application to path planning. *Proceedings of the European Control Conference*, page 5492–5499, 2007.

[55] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:1330–1334.

[56] K. Bolla, T. Kovács, and G. Fazekas. Trajectory building method for autonomous mobile robots. *Proceedings of TEAM 2014 Conference*, 4(1):170–173, 2014.

[57] R. O. Duda and P. E. Hart. Use of the hough transformation to detect lines and curves in pictures. *Comm. ACM*, 15:11–15, 1972.

[58] D.H. Ballard. Generalizing the hough transform to detect arbitrary shapes. *Pattern Recognition*, 13:111–122, 1981.

[59] C. R. Jung and R. Schramm. Rectangle detection based on a windowed hough transform. *Computer Graphics and Image Processing*, pages 113 – 120, 2004.

[60] C. R. Jung and R. Schramm. Parallelogram detection using the tiled hough transform. *Proceedings of 13th International Conference on Systems, Signals and Image Processing*, 2006.

[61] Open Source Computer Vision. OpenCV. `http://opencv.org/`, 2015. [Utolsó megtekintés: 2015-01-28].

[62] A. Pásztor, T. Kovács, and Z. Istenes. Compass and odometry based navigation of a mobile robot swarm equipped by bluetooth communication. *Proceedings of IEEE International Joint Conferences on Computational Cybernetics and Technical Informatics*, page 565–570, 2010.

[63] T. Kovács, A. Pásztor, and Z. Istenes. A multi-robot exploration algorithm based on a static bluetooth communication chain. *Robotics and Autonomous Systems*, page 530–542, 2011.

Saját publikációk listája

Könyvfejezetek

1. Bolla K., Kovács T., Fazekas G., Gathering of Fat Robots with Limited Visibility and without Global Navigation, Swarm and Evolutionary Computation, Springer, LNCS, Volume 7269, ISBN: 978-3-642-29352-8, ISSN: 0302-9743, pages: 30-38, 2012.

    - **Indexeli: DBLP, Springer, Scopus, ACM**
    - **Független hivatkozások száma: 8**

2. Bolla K., Kovács T., Fazekas G., Investigating the rate of failure of asynchronous gathering in a swarm of fat robots with limited visibility, Springer, LNAI, Volume 8468, ISBN: 978-3-319-07175-6, ISSN: 978-3-319-07176-3, pages: 249-256, 2014.

    - **Indexeli: DBLP, Springer, Scopus, ACM**

3. Bolla K., Johanyák Zs. Cs., Kovács T., Fazekas G., Local Center of Gravity Based Gathering Algorithm for Fat Robots, Springer, Issues and Challenges of Intelligent Systems and Computational Intelligence, Volume 530, ISBN 978-3-319-03206-1, pages: 175-183, 2014.

    - **Indexeli: Springer, Scopus**
    - **Független hivatkozások száma: 3**

Folyóiratcikkek

1. Bolla K., Kovács T., Fazekas G., A fast visual perception system based kin recognition and distance evaluation method in a robot swarm, GAMF Közlemények, ISSN: 1587-4400, pages: 49-61, Kecskemét 2010.

2. Bolla K., Kovács T., Fazekas G., A Barcode Based Kin Recognition and Identification Method for Robot Swarms, Scientific Bulletin of "Politehnica" University of Timişoara, BS-UPT TACCS Volume 56(70) No. 1 / March 2011., ISSN: 1224-600x, pages: 11-20, Temesvár, 2011.

3. Kovács T., Kovács L., Alvarez G. R., Bolla K., Csák B., Csizmás E., Drienyovszki S., Fábián Cs., Medgyes K., Osztényi J., Parameters of the Intelligent Driver Model in Signalized Intersections, Technical Gazette, IF.: 0.615, Eperjes, 2015. (megjelenés alatt)

    - **Impact Factor: 0.615**
    - **Indexeli: Scopus, Inspec, Compendex**

4. Bolla K., Johanyák Zs. Cs., Kovács T, Autonóm robotok gyülekezése lokális tömegközéppont alapú algoritmus használatával, GRADUS folyóirat, Volume 1, Number 1, ISSN: 2064-8014, pages: 137-144, Kecskemét, 2014.

Konferencia kiadványok

1. Bolla K., Kovács T., Fazekas G., A fast image processing based kin recognition method in a robot swarm, 1st Scientific and Expert Conference TEAM 2010, ISBN: 978-953-55970-0-2, pages: 165-170, Slavonski Brod, 2009.

2. Bolla K., Kovács T., Fazekas G., Compact Image Processing-based Kin Recognition, Distance Measurement and Identification Method in a Robot Swarm, ICCC-CONTI 2010 IEEE Conference, IEEE Catalog Number: CFP10575-CDR, ISBN: 978-1-4244-7431-8, Digital Object Identifier: 10.1109/ICCCYB.2010.5491237, pages: 419-424, Temesvár, 2010.

    - **Indexeli: IEEE Xplore, Scopus**

3. Johanyák Cs., Bolla K., Development of a Toolbox Supporting Fuzzy Calculations, International Symposium on Advanced Engineering & Applied Management - 40th Anniversary in Higher Education Conference, Volume 2, ISBN: 978-973-0-09340-7, pages: 155-158, Vajdahunyad, 2010.

4. Johanyák Cs., Bolla K., Fuzzy számításokat segítő eljárásgyűjtemény fejlesztése, TEAM/AGTEDU konferencia, ISSN: 1586-846x, Volume 2, pages: 437-442, Kecskemét, 2010.

5. Bolla K., Istenes Z., Kovács T., Fazekas G., A Fast Image Processing Based Robot Identification Method for Surveyor SRV-1 Robots, 2011 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM2011), IEEE Catalog Number: CFP11775-CDR, ISBN: 978-1-4577-0837-4 , ISSN: 2159-6247, Digital Object Identifier: 10.1109/AIM.2011.6027147 pages: 1003-1009, Budapest, 2011.

    - **Indexeli: IEEE Xplore, Scopus**
    - **Független hivatkozások száma: 1**

6. Bolla K., Kovács T., Fazekas G., Kiterjedéssel rendelkező autonóm robotok gyülekezése, AGTEDU 2012, pages: 129-137, Kecskemét, 2012.

7. Medgyes K., Bolla K., Csizmás E., Alvarez G. R., Kovács T., Fábián Cs., Osztényi J., Papp O., Performance of the wireless distribution system applied in a traffic intersection, TEAM 2013, pages: 248-252, Eperjes, 2013.

8. Kovács T., Kovács L., Alvarez G. R., Bolla K., Csák B., Csizmás E., Drienyovszki S., Fábián Cs., Medgyes K., Osztényi J., Mikroszkopikus közlekedési szimulátor fejlesztése és validálása, Közlekedéstudományi Konferencia, ISBN: 978-615-5298-30-1, pages: 173-181, Győr, 2014.

9. Bolla K., Kovács T., Fazekas G., Trajectory Building Method for Autonomous Mobile Robots, Proceedings of TEAM 2014, Volume 4, Number 1, ISBN: 978-615-5192-22-7, pages: 170-173, Kecskemét, 2014.

Absztrakt

1. Bolla K., Kovács T., Fazekas G., Distinguish competitor robot swarms with on-board visual perception system, CSCS PhD Conference, Szeged, 2010.

Egyéb

1. Bolla K., Robot azonosítással kibővített Surveyor SRV-1 robot szoftver, C programozási nyelv, 500 sor

2. Bolla K., Kovács T., MATLAB szimulációs program szinkron és aszinkron robot swarm kísérletekhez, MATLAB, 1500 sor

3. Bolla K., Autonóm mobil robotok beltéri navigációját segítő szoftver, C++ programozási nyelv, 4000 sor

Független hivatkozások listája

Saját cikkek és azok független hivatkozásai

Bolla K., Istenes Z., Kovács T., Fazekas G., *A Fast Image Processing Based Robot Identification Method for Surveyor SRV-1 Robots*, 2011 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM2011), IEEE Catalog Number: CFP11775-CDR, ISBN: 978-1-4577-0837-4 , ISSN: 2159-6247, Digital Object Identifier: 10.1109/AIM.2011.6027147, pages: 1003-1009, Budapest, 2011.

1. N. Mathews, G. Valentini, A. L. Christensen, R. O'Grady, A. Brutschy, M. Dorigo: *Spatially targeted communication in decentralized multirobot systems*, Autonomous Robots, Volume 38, Issue 4, DOI: 10.1007/s10514-015-9423-6, pages: 439-457, 2015.

* * *

Bolla K., Johanyák Zs. Cs., Kovács T., Fazekas G., *Local Center of Gravity Based Gathering Algorithm for Fat Robots*, Springer, Issues and Challenges of Intelligent Systems and Computational Intelligence, Volume 530, ISBN 978-3-319-03206-1, pages: 175-183, 2014.

1. R. Precup, T. Haidegger, L. Kovács, *Stable Hybrid Fuzzy Controller-based Architecture for Robotic Telesurgery Systems*, International Journal of Computational Intelligence and Pattern Recognition, Volume 1 (1), ISSN 0218-0014, pages: 61-76, 2014.

2. C. Pozna, R. Precup, P. Földesi, *A novel pose estimation algorithm for robotic navigation*, Robotics and Autonomous Systems, Volume 63, Part 1, DOI: 10.1016/j.robot.2014.09.034, pages: 10-21, 2014.

3. R. Precup, E.M. Petriu, L. Fedorovici, M. Radac, F. Dragan, *Multi-robot charged system search-based optimal path planning in static environments*, Intelligent Control (ISIC), 2014 IEEE International Symposium on, DOI: 10.1109/ ISIC.2014.6967643, pages: 1912-1917, 2014.

* * *

Bolla K., Kovács T., Fazekas G., *Gathering of Fat Robots with Limited Visibility and without Global Navigation*, Swarm and Evolutionary Computation, Springer, Lecture Notes in Computer Science, Volume 7269, ISBN: 978-3-642-29352-8, ISSN: 0302-9743, pages: 30-38, 2012.

1. C. Agathangelou, C. Georgiou, M. Mavronicolas, *A distributed algorithm for gathering many fat mobile robots in the plane*, Proceedings of the 2013 ACM symposium on Principles of distributed computing, ISBN: 978-1-4503-2065-8, DOI: 10.1145/2484239.2484266, pages: 250-259, 2013.

2. G. A. Di Luna, P. Flocchini, S. G. Chaudhuri, N. Santoro, G. Viglietta, *Robots with Lights: Overcoming Obstructed Visibility Without Colliding*, Lecture Notes in Computer Science, Volume 8756, 2014, pages: 150-164, 2014.

3. A. Honorat, M. Potop-Butucaru, S. Tixeuil, *Gathering fat mobile robots with slim omnidirectional cameras*, Theoretical Computer Science, Volume 557, DOI: 10.1016/ j.tcs.2014.08.004, pages: 1-27, 2014.

4. A. Pásztor, *Gathering simulation of real robot swarm*, Tehnical Gazette, ISSN 1330-3651 (Print), ISSN 1848-6339 (Online), pages: 1073-1080, 2014.

5. S. G. Chaudhuri, K. Mukhopadhyaya, *Leader election and gathering for asynchronous fat robots without common chirality*, Journal of Discrete Algorithms, DOI: 10.1016/ j.jda.2015.04.001, 2015.

6. S. Bhagat, S. G. Chaudhuri, K. Mukhopadhyaya, *Formation of General Position by Asynchronous Mobile Robots*, Cornell University Library, arXiv:1408.2072, 2014.

7. A. Pásztor, Zs. Czuprák, *Indirekt és direkt kommunikációt használó valós NXT robotok gyülekezési eljárásai*, Gradus, Kecskemét, 2014.

8. G.A. Di Luna, P. Flocchini, S. Gan Chaudhuri, F. Poloni, N. Santoro, G. Viglietta, *Mutual Visibility by Luminous Robots Without Collisions*, Cornell University Library, arXiv:1503.04347, 2014.