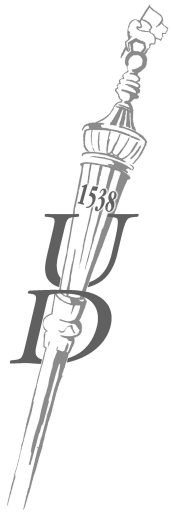
Egyetemi doktori (PhD) értekezés tézisei

Algoritmusok és implementációk a swarm intelligencia területén

Bolla Kálmán Milán

Témavezető: Dr. Fazekas Gábor



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

Tartalom – Contents

Bevezetés

Doktori disszertációm fő célja a mobil robotokkal végrehajtható feladatok vizsgálata és új megoldások kidolgozása volt. Kutatásaim során elsősorban mobil robot swarm-okkal foglalkoztam, ahol a feladatok végrehajtásához egyetlen robot helyett több robotot is felhasználunk. A swarm-ban található robotok egymással együttműködve, autonóm módon működnek, ahol minden egyed ugyanazokkal az érzékelőkkel és beavatkozókkal rendelkezik, és azonos ágensprogramot futtatnak. A feladatot végrehajtó robotok szerény számítási képeségekkel rendelkeznek, a cél elérése érdekében csak a feltétlenül szükséges érzékelőkkel és beavatkozókkal vannak felszerelve, így a kialakításuk miatt olcsók és könnyen pótolhatóak. Amennyiben egy vagy több egyed időközben üzemképtelenné válik, a feladat végrehajtása nem kerül veszélybe, hiszen a többi robot nélkülük is képes elérni a kitűzött célt. Többágenses környezetben szükségessé válik a társak biztonságos felismerése és azonosítása. Erre a problémára dolgoztam ki egy olyan vizuális érzékelésen alapuló eljárást, melynek segítségével a robotok fel tudják ismerni a swarm-ban található társrobotokat. Megoldásom továbbá felhasználható a megfigyelőtől való távolság becslésére is, ami szintén egy fontos kritériuma a swarm algoritmusok végrehajtásának. Az eredeti eljárás kisebb módosításával lehetőség nyílik a társak azonosítására is, így azon felül, hogy a megfigyelő meg tudja becsülni a számára látható robotok relatív pozícióját, az egyes robotokat képes identifikálni.

Mobil robot swam-okkal számos feladatot végrehajthatunk, ezek közül a szakirodalom három alapvető tevékenységet határoz meg: egy pontba gyülekezés, csoportosulás, és mintázat kialakítás. Ezen feladatok közül kuta-

tásaim során a gyülekezés feladatát vizsgáltam meg tüzetesebben, melynek definíciója a következőképpen hangzik: a swarm-ban található robotoknak véges idő alatt egy pontban kell találkozniuk. Természetesen ezt a feltételt csak akkor lehet teljesíteni, ha a robotokat pontszerűnek tekintjük. A gyülekezés sikerességét alapvetően az befolyásolja, hogy a robotok milyen képességekkel rendelkeznek. Amennyiben az egyedek fel vannak szerelve globális érzékelő rendszerrel és mindenki tud kommunikálni közvetlen vagy közvetett módon a társaival, a gyülekezés egy triviális feladattá válik, mivel az egyedek megbeszélnek egy pozíciót, ahol találkozniuk kell és a feladat ezzel meg is oldódott. Ezért fontos azon minimális képességek meghatározása, melynek esetében a gyülekezés problémája még megoldható, de nem lesz triviális. Ez esetben olcsóbb kialakítású robotok segítségével is el tudjuk érni ugyanazt az eredményt. Mindamellett fontosnak tartom megemlíteni, hogy a swarm működése lehet szinkrón és aszinkrón. Szinkrón működés alatt azt érjtük, hogy minden aktivációs ciklus egyszerre kezdődik el minden egyes robotnál. Aszinkrón működés esetében minden robot életciklusa attól függően rövidebb vagy hosszabb, hogy mennyi idő alatt képes elérni az aktuális részcélját. A swarm-ban található robotokat tekinthetjük mind pontszerűeknek, mind kiterjedéssel rendelkezőeknek. A kiterjedéses robotreprezentáció esetében már lehetségessé válik a gyülekezés valós, fizikai robotokra való adaptációja is. Ebben az esetben a robotok nem foglalhatják el ugyanazt a pozíciót, mint a pontszerű robotoknál, valamint figyelembe kell venni azt is, hogy az algoritmus végrehajtása során ütközések is történhetnek. Természetesen így az eredeti gyülekezési definíció is megváltozik, a gyülekezés akkor fejeződik be, ha kialakul az úgynevezett kontakt gráf. A kontakt gráfban a robotok olyan közel helyezkednek el egymáshoz, hogy egymást érintik vagy egy meghatározott méretű biztonsági sugáron belül helyezkednek el.

Gyülekezéssel kapcsolatos kutatásaim során megvizsgáltam a szakirodalomban található megoldásokat és kidolgoztam egy saját szinkrón működésű eljárást. Ezzel egy időben szimuláció segítségével összehasonlítottam egy, már létező algoritmussal saját megoldásomat, majd meghatároztam azokat a minimális képességeket, melyekkel még lehetséges a gyülekezés megvalósítása. A szimulációk során kiterjedéssel rendelkező robotokat használtam,

melyek limitált látótávolsággal rendelkeznek. A valós robotreprezentáció miatt ügyelni kellett az ütközések kezelésére is, így egy ütközéselkerülő algoritmus is kidolgozásra került. A szimulációk során elsősorban a gyülekezés végrehajtásához (kontakt gráf létrejöttéhez) szükséges időt és a létrejött alakzat tömörségét mértem és hasonlítottam össze. Egy másik publikációmban egy ismert szinkrón működési elvű gyülekezési algoritmust vizsgáltam aszinkrón végrehajtás során. Célom az volt, hogy aszinkrón gyülekezés sikerességének valószínűségét mérjem különböző számítási teljesítmények mellett.

Automón mobil robotok egyik legnagyobb kihívásokkal rendelkező tevékenysége a navigáció. Ezt az alapvető tevékenységet minden mobil robotokkal végrehajtott feladatnál használjuk. A navigáció pontosságát számos tényező akadályozza, amitől a feladat nem triviális. A nehézséget tulajdonképpen az érzékelőkből és a beavatkozókból származó hiba adja, mivel egy rövidebb trajektória megtétele után is rengeteg hiba halmozódik fel, így a robot nem abba a pozícióba kerül, ahová azt ő előzetesen feltételezi. Az érzékelőkből adódó hibát két csoportba sorolhatjuk: szisztematikus és nem szisztematikus hibákról beszélhetünk. Előbbi minimálisra csökkenthető a rendszer kalibrációjával, így például egy egyszerű odometriás eljárással is pontos navigációt érhetünk el. Hátránya azonban, hogy robotonként, azon belül is érzékelőnkként kell végrehajtani a kalibrációt, ami például egy fizikai robotokkal végrehajtott swarm kísérlet esetében hosszú előkészítést jelent. Kutatásaim során egy saját kamera érzékelőt és jellemző környezeti objektumokat használó eljárást dolgoztam ki és vizsgáltam annak pontosságát. Algoritmusom működésének helyességéhez számos kísérletet végeztem és az eredményeket összehasonlítottam egy jól bekalibrált odometriás érzékelés hibájával.

Az értekezés új tudományos eredményei

1. Társak felismerése és azonosítása mobil robot swarm-okban

1.1. Társfelismerés

Tézis 1: *Sikerült egy olyan társfelismerő megoldást kidolgozni mobil robot swarm-okhoz, mely a robotokra elhelyezett vizuális érzékelő rendszer és egy speciális jelölő mintázat alapján meg tudja különböztetni a swarm-ban található egyedeket a környezet további objektumaitól.*

Az általam kifejlesztett társfelismerő eljárás képfeldolgozáson alapuló módszer, ezért feltételezem, hogy a swarm-ban található robotok fel vannak szerelve képalkotó rendszerrel. Kamera által rögzített szürkeárnyalatos (intenzitás) 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 látszódni az algoritmus bemeneteként használt rögzített szürkeárnyalatos képen, ezért a mintázat fekete és fehér színek ismétlődéséből áll. További fontos kritérium volt, hogy a robotok külső munkaállomás nélkül, önállóan képesek legyenek a rögzített képeket feldolgozni. Tehát az eljárást úgy kellett megalkotni,

hogy az egyedek gyenge számítási képessége ne jelentsen problémát, és ezáltal autonóm viselkedést érjünk el.

A kamera által rögzített szürkeárnyalatos jelenetet oszloponként mintavételezzük, mivel feltételezzük, hogy a robot swarm tagjai sík terepen dolgoznak. Így a minta esetleges elforgatottjaival nem kell foglalkozni, kevesebb számítást kell végrehajtani. A mintavételezés után a kiválasztott oszlopon végrehajtjuk az egydimenziós gyors Fourier-transzformációt és a továbbiakban a frekvenciatérben keressük a mintázat meglétét. Mivel a társfelismeréhez használt ismétlődő mintázat egy frekvenciát kódol, a frekvenciatérben egy jól érzékelhető lokális szélsőérték jelenik meg, ha az adott oszlop része a mintának. Kísérletek segítségével megállapítottam, hogy a társfelismeréshez használt mintázat különböző távolságokban mekkora frekvenciát generál, és egyúttal meghatároztam azt a maximális távolságot, ahol még biztonságosan használható a társfelismerés VGA felbontás mellett. Végül a frekvenciatartományban egy küszöbfüggvénnyel döntöttem el, hogy az adott oszlop a mintázat része vagy sem.

Mintavételezett oszlop feldolgozása után a teljes mintázat vagy akár mintázatok meghatározása a cél a rögzített digitális képen (attól függően, hogy hány robot található a kamera látószögében). Az egyes jelenetek feldolgozásának eredményei nem befolyásolják a jövőbeni képek feldolgozását, tehát a robot semmilyen információt nem használ a társak kereséséhez az aktuális jeleneten kívül. A teljes kép feldolgozása helyett kihasználtam a mintázat fizikai méreteit és egy mintavételezési konstans segítségével választottam ki a vizsgálandó oszlopokat. Amennyiben nincsen képen egyetlen társrobot sem, ebben az esetben nem kell az egész jelenetet oszloponként feldolgozni és az algoritmus futási ideje is kevesebb lesz. Ha találunk egy a mintázathoz tartozó oszlopot, akkor annak bal-, illetve jobbszomszédságát elemezve megkapjuk a teljes mintázatot. A kidolgozott módszer működését Surveyor SRV-1 robotok segítségével ellenőriztem. Első változatban a robotok külső számítógéppel kommunikálva dolgozták fel a képeket, tehát a robotok csak a képeket rögzítették, majd továbbították egy külső feldolgozó egységnek, ahol egy MATLAB program feldolgozta a rögzített képeket. A feldolgozást követően a robotok visszakapták a külső számítógéptől a tár-

sak relatív pozícióját. Későbbiekben magára a robotra is implementáltam az algoritmust, így teljesen autonóm viselkedést kaptam.

1.2. Társak azonosítása

Tézis 2: *Társak azonosítása lehetséges a társfelismerő algoritmus és a felhasznált mintázat továbbfejlesztésével. Továbbá az azonosítás megoldást ad a robotok megfigyelőtől vett félig takarására is.*

A robotok azonosításához módosítani kell a meglévő ismétlődő mintázatot, ami végső soron egy vonalkódnak fog megfelelni. Az eredeti mintázatban egy vagy több hibát ejtünk, amivel a társfelismerő algoritmus továbbra is biztonságosan működik, egyúttal az azonosítás is lehetségessé válik. Mintázaton végzett módosítások kombinációjával tudunk annyi egyedi azonosítót előállítani, amennyi a swarm összes robotjának szükséges.

Frekvenciatartomány helyett az azonosítást képtartományban végeztem a képen található textúra primitívek felhasználásával. A mintázat tulajdonságainak felhasználásával lehetőségem nyílt egy megbízhatóan működő azonosító eljárást kialakítani. Ebben az esetben sem kellett a mintázat elforgatottjaival foglalkozni, mivel feltételeztem, hogy a robotok egyenes padlós környezetben dolgoznak. A társfelismerés által kiválasztott oszlopokat az azonosítás során egyesével dolgoztam fel. Első lépésként él-kiemelést végeztem Sobel él-detektálóval, él-lokalizációt pedig küszöbölés segítségével. Mivel a mintázatot alkotó elemek fehér és fekete színűek (továbbá a megvilágítási viszonyokat az egész munkaterületen egyenletesnek tekintettem), ezért elegendő volt egyetlen fix küszöbérték használata minden él-lokalizációnál. Küszöbölés után kiszámoltam az egyes szegmensek szélesség tulajdonságát is. A felhasznált ismétlődő mintázat, alapvetően csúcsok, völgyek és textúra primitívek ismétlődéséből áll, ahol a teljes mintázatot fix számú szegmens alkotja. Az algoritmus először az összes lehetséges, a felhasznált mintázattól függő ismétlődést tartalmazó részt határozza meg a küszöbölt eredményeken, majd a szélesség tulajdonságok alapján dolgozik tovább. Minden lehetséges esetre kiszámolja a szélesség tulajdonságok 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. A mintázat oszlopon belüli elhelyezkedése után már lehetővé válik a textúra primitív alapú azonosítása, így minden társfelismerés során kiválasztott oszlop feldolgozása után megkapjuk, hogy a képen milyen azonosítójú robotok találhatóak.

Amennyiben az összes társfelismerés alapján megjelölt oszlopot feldolgoztuk, megoldást kapunk a félig takarás problémájára is. Mivel a robotok egymás előtt, illetve mögött helyezkedhetnek el a megfigyelő szempontjából, a rajtuk elhelyezett mintázatokat a társfelismerő eljárás egynek tekintheti. Oszloponként végighaladva az érzékelt mintázaton minden oszlophoz rendelünk egy-egy robot azonosítót, amelyből egyértelműen eldönthető, hogy a mintázat egy vagy több robothoz tartozik-e.

1.3. Megfigyelőtől való távolság becslése

Tézis 3: *Megfigyelőtől való távolság megbecsülhető a mintázat által generált frekvenciából.*

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 alacsonyfrekvenciás régióban keletkezik, viszont ha távol helyezzük el, magasabb frekvenciát generál. A módszer alkalmazása különösebb plusz számítást nem igényel, hiszen a detektált frekvencia már a társfelismerés után rendelkezésre áll. Pontosabb eredmény érdekében a teljes, felismert mintázat által generált frekvenciák átlagával érdemes számolni, így csökkenthető az érzékelésből származó hiba. Az elvégzett mérések és kísérletek alapján arra a megállításra jutottam, hogy a frekvenciából becsült távolságmérés ± 10 cm-es pontossággal működik, 50 cm-től 300 cm-ig bezárólag.

1.4. Megjelent publikációk

A fejezetben bemutatott eredményekhez kapcsolódóan több publikációm jelent meg folyóirat- és konferenciacikkek formájában (összesen 6 darab). Ezekből 2 angol nyelvű folyóiratcikkben ([1], [2]), 3 konferencia kiadványban megjelent angol nyelvű cikkben ([3], [4], [5]) és egy absztraktban ([6]) közöltem az elért kutatási eredményeimet. [5] cikkemre már van egy független hivatkozásom, kettő pedig ([4], [5]) megtalálható a Scopus adatbázisban.

2. Gyülekezési algoritmusok globális információk nélkül

2.1. Saját szinkrón működésű gyülekezési algoritmus

Tézis 4: *Kidolgoztam egy olyan szinkrón működésű gyülekezési algoritmust, amely sikeresen megoldja a gyülekezés problémáját véges idő alatt, kiterjedéssel rendelkező mobil robot swarm-ok esetén.*

Saját gyülekezési algoritmusom szinkrón működést feltételez, ahol a robotok kiterjedéssel rendelkeznek és nem átlátszóak. Továbbá a gyakorlati megvalósíthatóság szempontjából a robotoknak limitált a látótávolsága és nincsenek globális navigációs rendszerrel felszerelve, emellett nem rendelkeznek memóriával és az egyes egyedek nem azonosíthatóak. Célom az volt, hogy a felsorolt minimális tudással is végrehajtható gyülekezési algoritmust készítsek.

Az algoritmus indításakor feltételezem, hogy a láthatósági gráf összefüggő, amely alapfeltétele a limitált látótávolságú mobil robotok sikeres gyülekezésének. A gyülekezés végrehajtása közben a láthatósági gráfnak végig összefüggőnek kell lennie, ellenkező esetben több csomópont jön létre, a gyülekezés pedig sikertelen lesz. 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 nevezzünk minden olyan egyedet, mely legfeljebb 120°-os szögben lát szomszédos robotokat (tehát a swarm külső határán helyezkedik el), a többi robotot pedig belső robotnak tekintjük. A robotok halmazát minden egyes végrehajtási ciklusban fel kell bontani az előbb említett két részhalmazra.

Gyülekezési algoritmusom célja a következő: a periméter robotokat szeretnénk a lehető leggyorsabban a belső robotok felé mozdítani. Önmagában az algoritmus működése azt eredményezné, hogy a láthatósági gráf felbomlik, és különböző csomópontok alakulnak ki. Azonban a belső robotok mozgásának lassításával és a célvektort korlátozó algoritmus segítségével [7] ez a probléma kiküszöbölhető.

A kiterjedéssel rendelkező robot reprezentáció hátránya, hogy a robotok akadályozhatják egymást mozgás közben, így a robotok úgynevezett holtpont helyzetbe kerülnek. Ennek feloldása egy egyszerű módszerrel megvalósítható: a blokkolt robot eredeti célvektorát úgy kell módosítanunk, hogy érintő irányba történjék az új elmozdulás, így a robot közelebb kerül az eredeti célhoz és a blokkoló robottól is megfelelő távolságra távolodik el. A módszert csak akkor alkalmazzuk, ha egy blokkoló robotunk van, több blokkoló esetében nem, mivel lehetséges, hogy az aktuális robot már elérte a végső pozícióját.

Algoritmusom helyes működésének tesztelése érdekében MATLAB szimulációt használtam. Saját megoldásomat összehasonlítottam egy, az irodalomban megtalálható, hasonló feltételek mellett működő eljárással ([7]), illetve annak egy módosított változatával. Vizsgáltam a végrehajtáshoz szükséges időt, valamint a létrejött alakzat tömörségét. Az eredményekből egyértelműen látszott, hogy a gyülekezéshez szükséges idő mind a három esetben nagyon hasonló volt, viszont a létrejött alakzat sokkal kisebb átmérőjű lett saját megoldásom esetében. Tehát az új algoritmus tömörebb alakzatba gyűjtötte össze a swarm-ban található robotokat.

2.2. Szinkrón gyülekezési algoritmusok alkalmazása aszinkrón esetben

Tézis 5: *Ando, Suzuki és Yamashita szinkrón működésű és pontszerű robotreprezentációjú gyülekezési algoritmusa teljesíti a gyülekezést aszinkrón működés és kiterjedéses robotreprezentáció esetén is, amennyiben a társak érzékeléséhez és a következő pozíció számolásához szükséges idő konvergál a nullához.*

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 az összes robotnál. A probléma alapvetően az érzékelés és az ahhoz szükséges idő, valamint a számítási teljesítményből adódik. Ezért feltételeztem, ha a *Look* és a $Calculate$ műveletek végrehajtásához szükséges idők közelítenek a 0-hoz, akkor a megszakadások átlagos gyakorisága is csökkenni fog. Célom tehát az volt, hogy szimulációs kísérletekkel meghatározzam a megszakadás valószínűségét meghatározott méretű swarm mellett. Ebben az esetben is minimális robot tulajdonságokat és kiterjedéses robotokat feltételeztem végig a kísérletek alatt.

A szimulációk során a társak érzékelésére a korábban bemutatott, saját társfelismerő és távolság becslő eljárásomat használtam. Ezzel a módszerrel képesek vagyunk megbecsülni a társak relatív pozícióját a környezetben egyetlen vizuális érzékelő segítségével. Az volt a feltételezésem, hogy a robotok omnidirekcionális képet képesek rögzíteni, és minden robot rendelkezik a társfelismeréshez szükséges mintázattal. A szimuláció működtetéséhez először meg kellett állapítanom a *Look* fázisban szükséges feldolgozási időket, amihez a Surveyor SRV-1 robotra implementált társfelismerő eljárással végeztem kísérleteket. A láthatósági sugár mértékét is az előbb említett eljárás adta, valamint minden más paramétert is ettől a robottípustól vettem át (robot sugara, maximális utazási sebesség).

A szimulációk megkezdése előtt a kezdőpozíciókat kellett létrehozni, ahol a láthatósági gráfnak összefüggőnek kellett lenni. Négyzet alakú környezetben generáltam egyenletes eloszlással a robot pozíciókat, egy előre

meghatározott robotsűrűséggel (1/9 robot/m^2). Ezt a sűrűséget felhasználva a kapott átlagos robot-robot távolságok közel lesznek a láthatósági sugár (V) mértékéhez. A generálás után kiválasztottam a legnagyobb összefüggő gráfot, amely a bemenete lesz a felhasznált gyülekezési algoritmusnak. Az eljárás egyenes következménye, hogy nem lehet előre megmondani pontosan a swarm tagjainak a számát (N). Erre a problémára egy lehetséges megoldásunk van, ha különböző intervallumokat határozunk meg konkrét robotszámok helyett. Ezek a következőek 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.

Szimulációk során Ando [7] szinkrón gyülekezési algoritmusát használtam aszinkrón feltételek mellett, [8] publikációban is erre az algoritmusra bizonyítottak be egy speciális esetet, amikor a láthatósági gráf megszakad. Összesen 3600 kísérletet végeztem MATLAB szimulációk segítségével, négy különböző 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, amely alatt a robot érzékeli a környezetét és meghatározza a társak helyzetét a korábban részletezett képfeldolgozási eljárás segítségével. 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, illetve egy megfelelően nagy számítási teljesítmény esetén közelíteni fog a 0-hoz. Fontos, hogy minden egyes szimuláció esetében a robot maximális utazási sebességét használtam, viszont ugyanezeket az eredmények kaptam volna 5-ször, 10-szer és 20-szor lassabb robotmozgások esetén is.

Összesen két leállási feltételt határoztam meg. Az első természetesen az, amikor a swarm sikeresen összegyűlt, tehát a kontakt gráf létrejött. Második leállási feltételnek pedig a láthatósági gráf megszakadását adtam

meg. Minden egyes robot e két feltétel bekövetkezéséig folytatja a *Look* – *Calculate* – *Move* ciklus végrehajtását.

Az eredményekből arra a következtetésre jutottam, 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 pontszerű robotok esetében a swarm nagyságrendekkel többször szétszakad, mely 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 a SEC alapú algoritmusnak [7] is, ami a látható szomszédokra húzott legkisebb kör középpontja felé mozdítja a robotot.

2.3. Megjelent publikációk

Gyülekezési algoritmusok vizsgálatával kapcsolatban 3 könyvfejezetem ([9], [10], [11]) jelent meg, melyek megtalálhatóak a Scopus adatbázisban. Továbbá ebből kettő DOI azonosítóval rendelkezik és megtalálható a DBLP adatbázisában is. [9] cikkemre jelenleg 5, [11]-re pedig 3 független hivatkozásom van. Ezenfelül egy magyar nyelvű folyóiratban ([12]) és egy konferenciacikkben ([13]) foglaltam össze az elért eredményeimet.

3. Beltéri navigáció támogatása képi információk alapján

Tézis 6: *Sikerült kifejleszteni egy vizuális érzékelőn alapuló, mobil robot beltéri navigációt segítő eljárást, amelynek pontossága megközelíti egy jól bekalibrált odometriás eljárás pontosságát.*

Az általam kidolgozott eljárásban a környezetben elhelyezett jellemző objektumok 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 fel-

használható legyen. Ez alatt az értem, hogy az odometriás eljárásnál több robot egyidejű használata esetén a robotokat egyesével be kell kalibrálni, ezzel ellentétben saját megoldásomban 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 robotunk a vizuális érzékelő rendszerét fogja használni a helymeghatározáshoz. Feltételezzük, hogy a robot olyan beltéri környezetben dolgozik, amely sík felülettel rendelkezik, és a felület 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 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, továbbá a mintázat alapján a robot fel tudja térképezni környezetét is.

A jelenet előfeldolgozását szürkeárnyalatos képpé alakítással kezdem, majd adaptív küszöböléssel elválasztom a hasznosnak ítélt pixeleket a háttér pixeleitől. Tehát feltételeztem, hogy a kép előtér és háttér objektumai intenzitás alapon egymástól elválaszthatóak. Adaptív küszöbölésre azért volt szükség, mert a megvilágítási viszonyok és a robot által vetett esetleges árnyék miatt egyetlen globális értékkel nem lehetett a küszöbölést biztonságosan végrehajtani. A küszöbölés után tapasztalható volt a bináris képen némi só- és borsszerű zaj, aminek csökkentésére medián szűrőt használtam. Következő lépésként az előre kiszámolt perspektivikus transzformációval a torzítatlan felülnézeti képet állítom elő, így a későbbiekben már a párhuzamosságot és a merőlegességet is figyelembe tudom venni a teljes mintázat felismeréséhez. Az előfeldolgozott képen látható egyeneseket alkotó pixelekből Hough-transzformációval állapítom meg az egyeneseket leíró geometriai információkat, melyeket polár koordinátában tárolok. A

Hough-transzformáció alkalmazása során a beállított alacsony küszöb miatt sok egyenes jelöltet kaptam, ami nagyban befolyásolta a további eljárások működését. Így szükségessé vált az egyenes jelöltek klaszterezése és az egyes klaszterekben található, a pixelekre legjobban illeszkedő egyenesek kiválasztása.

A teljes mintázat felismerése az előbbiekben megállapított egyenes primitívek alapján történt. Első lépésként párhuzamos párokat kerestem, majd megvizsgáltam minden párhuzamos pár egymással való merőlegességét. Itt figyelembe kellett venni a környezetet borító mintázat tulajdonságait is, illetve hogy az egymásra merőleges párok ne legyenek sem túl közel, sem túl távol egymástól. Ezzel a módszerrel kiszűrhetőek a hibás érzékelések is, mivel csak a mintázatot alkotó objektumokat kapjuk meg az eljárás végén.

Jellemző objektumok követéséhez fel kellett címkézni a mintázat minden egyes primitívjét, és a további jeleneteken meg kellett valósítani azok ismételt azonosítását is. Az objektumkövetéshez távolság metrikát használtam, ahol figyelembe kellett venni a jelenlegi és a múltbéli érzékeléseket. Szükséges volt továbbá a hibás érzékelések kezelésére tárolni azt, hogy hány jeleneten keresztül nem azonosítottuk be az adott objektumot, illetve összesen hány alkalommal volt megtalálható a robot által rögzített képeken.

A robot egyik feladata, hogy az ágensprogram végrehajtása közben bizonyos időközönként eltárolja a két érzékelés közötti elmozdulás vektorát annak érdekében, hogy felépítse a megtett utat reprezentáló trajektóriát. 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, melyeket egy előre meghatározott küszöbnél többször megtaláltunk már a korábbi jeleneteken. Minden mintavételezésnél a számolt vektorok átlagát tároljuk, így végül a mentett vektorokból fog állni a trajektória, ami a robot által megtett utat írja le.

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 [14], [15], [16] cikkekben leírt kalibrált odometriás eljárással hasonlítottam össze, ahol a relatív hiba 2.0E-3 volt. A relatív hibát $\Delta L/L$ –el definiáljuk, ahol ΔL jelenti a robot pozíciójának hibaátlagát, miután a robot megtett egy L hosszúságú utat [17]. Odometria esetén ez a relatív hiba konstans lesz, tehát a megtett úttal együtt a pozíció becslésében a bizonytalanság mértéke nőni fog.

A kísérletek beállításánál azt feltételeztem, hogy a robot egyenes vonalú egyenletes mozgást végez 20 m hosszúságú szakaszon. 2 és fél méterenként rögzítettem melyik kameraframe-nél tart a felvétel, így a teljes 20 m-es szakaszon belül 8 mérföldkövet határoztam meg, melyek a későbbiekben fontos szerepet töltenek be a pontosság mérésében. Összesen 15 tesztszakasz került felvételre, melyeket utólag dolgoztam fel az e célra kifejlesztett feldolgozó programommal. A kísérletek közül 10 kísérlet szolgált az algoritmusom kalibrálására, a fennmaradó 5 pedig a pontosság ellenőrzésére. A kísérletek eredményein az volt látható, hogy a relatív hiba kisebb megtett távolság esetén nagyobb volt, mint az odometriás eljárásé. Viszont a megtett út növekedésével a relatív hiba csökkent, végül a kalibrált odometria pontosságát először megközelítette, majd a 2.0E-3 relatív hibahatár alá ment.

3.1. Megjelent publikációk

Beltéri navigációt segítő eljárásomat egy angol nyelvű konferenciacikkben [18] publikáltam.

* * *

Introduction

The main goal of my dissertation is to examine and create new algorithms for mobile robots. My research mainly focuses on the area of mobile robot swarms, where the tasks are executed by more mobile robots instead of one. The robots work together in the swarm and execute their tasks autonomously, and each unit is equipped with the same kind of sensors and effectors and runs the same agent program. These robots have limited computational capacities, and in order to reach their goals they are equipped with the necessary sensors and effectors, and due to these they become cheaply designed and easily replaceable. In the case of one or more units break down, the other robots are still able to achieve the goals so the task of the whole swarm will not be risked. In a multi-agent environment a reliable recognition of the members is necessary. With regards to this problem I have developed a method based on visual perception, which enables each robot to recognise the other members of the robot swarm. On the other hand my method is also suitable for estimating the distances between the observers and the recognised robots, which is another criterion of successfully executing tasks in the robot swam. If we want to upgrade the original recognition method of the swarm members, it is also possible to identify the member robots of the swarm.

Mobile robot swarms are suitable for completing numerous tasks but the researchers focus on three basic tasks such as: gathering, flocking and pattern formation. During my research I was closely analysing the gathering problem, which is defined the following way: the swarm members must meet in a single point within a finite time. In order to fulfil the constraint of the

single point gathering the robots are regarded as points. The success of the gathering is essentially influenced by the capabilities of the robots. If global navigation is available and the members are able to communicate in direct or indirect ways, the gathering becomes a trivial task as the robots are able to agree on the encounter point and thus the gathering problem is solved. The next step would be to determine the minimal capabilities of the robots that are still capable to solve the gathering, although the problem remains it is not trivial. As a consequence of this the gathering will be solvable with the help of cheaper robot constructions. Nevertheless, the agents in the swarm could perform their activities synchronously or asynchronously. Regarding the whole robot swarm in the synchronous case each activity cycle starts at the same time. On the other hand in the asynchronous case the required time of executions of the activities could be shorter or longer in accordance with each robot. The robots could be represented as points or with their extension (also called "fat robots"). Fat robot representation allows adapting the gathering for real, existing robots as well. In this case the robots cannot occupy the same position whereas in the point robot representation collisions are possible among robots as well. Due to the fat robot representation the definition of the original gathering must be adjusted: the gathering is completed when the contact graph occurs. In the contact graph the robots are situated close to each other, or can even touch one another, or can be situated in a safety radius of another swarm member.

During my doctoral research I have studied the available solutions in the literature and I have developed my own synchronous gathering method. I have tested my approach with simulations and I have compared it with the previously existing gathering algorithms. At the same time the minimal capabilities of the robots are determined, which allows us to complete the gathering successfully. During the process of the simulation I used the fat robot representation and robots with limited visibility. Due to the real robot representation the possibilities of collisions and deadlock situations had to be taken into consideration, thus the collision and deadlock solving algorithm was developed. During the simulations the necessary time for the execution and the compactness of final swarm shape were also investigated.

In another project, I tested a well-known synchronous gathering algorithm along with asynchronous setting. My goal was to investigate the gathering rate of failure while I was using different computational capable robots.

Navigation is the most challenging task for a mobile robot. This basic operation is part of every kind of robot tasks. The accuracy of navigation can be influenced by several parameters, and due to this the navigation becomes a less trivial task. The difficulty originates from the sensors and effectors since a short trajectory taken by the robot is affected by errors, thus the robot will be situated not in the same position that it was expected. Errors from sensors can be classified as systematic and non-systematic errors. The effect of systematic errors can be decreased by the calibration of the system so the odometer based navigation becomes fairly accurate. The main drawback of odometer calibration is that the calibration must be executed on each robot sensor, which means that preparing a mobile robot swarm experiment becomes time-consuming. The main goal of my research is to develop an accurate camera-based navigation method that uses visual markers in its environment. The proposed method has been tested and compared to the accuracy of a well-calibrated odometer-based solution.

New results

1. Recognition and identification of mobile swarm members

1.1. Swarm member recognition

Thesis 1: *I have successfully developed a swarm member recognition method, which is able to distinguish the robots from other environmental objects with the help of a visual perception system and a special robot marker.*

My robot recognition method is based on image processing, assuming that each robot is equipped with a visual perception system. The intensity image, which is captured by the camera is used to define the relative positon of the swarm members and other environmental objects. The recognition method is based on a periodic pattern placed on each robot. This pattern must properly be visible in the captured intensity image so the pattern is in black and white colour. Another important criterion is the autonomous behaviour of the members thus the robots have to process the captured frames within their own hardware. For this reason my robot recognition algorithm is prepared to run properly in case of weak computational performances as well, which would enable the robots to work autonomously.

The captured intensity image is sampled by columns as I assume that the environment is a flat ground. In this case the rotations of the pattern are not taken into consideration and less calculation will be executed. After the sampling, the chosen column is processed by the fast Fourier transform and

the existence of the robot pattern is inspected within the frequency domain. Since the periodic pattern encodes a frequency, a peak (local maxima) appears in the frequency domain if the sampled column goes through the pattern. In order to test my recognition method I have performed several experiments to reveal the connections between the detected frequencies and the distances. Moreover, these experiments have been suitable for determining the maximum distance, in which my recognition method works reliably with VGA camera resolution. Eventually a threshold function is used to separate columns from the parts of the pattern and other parts.

After the sampled column is analysed, the whole pattern or patterns (it depends on the number of visible robots) must be processed in the captured image. Note that the previous images do not influence the current or future detections, thus there is no available a priori information for the current image processing. In order to process the whole image properly, the physical dimensions of the pattern are employed, and a sampling constant is introduced to pick a column from the image. If the image does not contain any visible robots, the whole frame does not sample by columns so the sampling constant allows decreasing the algorithm execution time. If a column of the pattern is found, the neighbouring columns are analysed in order to get the whole pattern. I tested my method with the help of Surveyor SRV-1 robots. In the first version of my algorithm the robots communicated with an external PC, which processed the captured images with help of a MATLAB program. After processing the PC sent the relative positions of the surrounding robots back to the client robot. In the final version the robots ran the whole processing method in their agent program so they worked autonomously.

1.2. Identifying swarm members

Thesis 2: *The identification is possible with the use of an upgraded periodic pattern and the swarm recognition method. Moreover, the identification provides a solution to the robot partially covered by each other from the angle view of the observer.*

In order to identify the periodic pattern we must upgrade it to a barcode-like pattern. One or more defected part can be placed into the original pattern and the identification will be possible and the recognition algorithm will work. The combination of the defected parts provides all the robots in the swarm with the necessary unique identifiers.

Instead of using the frequency domain again, with the help of texture primitives the spatial domain is applied to identify the barcode pattern. The properties of the new pattern enable us to create a robust identification method. In this case, due to the flat ground environment the rotations of the barcode pattern can be discarded as well. Only the previously selected columns are processed by the identification algorithm. The first step is to enhance the edges by the Sobel detector and localise them with thresholding. Since the pattern contains black and white parts (assuming that the lighting conditions are the same in the whole environment) it is possible to use only one threshold value to solve the edge localisation. The width and height properties of each segment are calculated after the thresholding. The applied pattern is simplified into repetition of peaks, valleys and texture primitives, and the entire pattern has a fix number of segments. The proposed algorithm determines all possible places of the pattern and it uses the width properties for further work. The variance of pattern candidates is calculated based on the width property. The place of the pattern is where the variance is the minimum, which means the edges are located the same distance from each other. After the vertical location of the pattern is determined, it is possible to identify the pattern by the texture primitives. Eventually, all processed columns provide the identified robots in the captured image.

As long as the columns are selected by the recognition method and are processed with the identification algorithm, the partially covering problem is solved as well. Since the robots are situated behind or in front of each other from view angle of the observer, the original robot recognition method considers two robots as one when they partially cover each other. After identification, it becomes clear which column belongs to which robot.

1.3. Estimating robot-robot distances

Thesis 3: *The observer robot is able to estimate robot-robot distances from the detected frequency.*

One of the most important benefits of Fourier analysis is that the observer robot is able to estimate the relative distance between the observer and the recognised robots using a camera frame. Due to this, if we place the pattern close to the observer, it generates a frequency in the lower frequency region, however if the pattern is far from the observer, the peak appears in the higher frequency region. This method does not need any notable additional calculation since the detected frequencies are available after the swarm member recognition. The average of detected frequencies can be used in order to get a more accurate result which allows for decreasing the errors. According to the accomplished measurements, the frequency-based distance estimation is able to estimate the distance with ± 10 cm uncertainty from 50 cm to 300 cm.

1.4. Publications

In connection with the introduced results of this chapter several publications have been appeared in journals and conference proceedings (6 pieces). I have published 2 journal articles ([1] [2]) and I have 3 conference papers ([3] [4] [5]) and an abstract ([6]) written in English. For my [5] article I already have an independent reference.

2. Gathering algorithms without global information

2.1. The proposed synchronous gathering algorithm

Thesis 4: *I have prepared a synchronous gathering algorithm, which successfully fulfil the gathering within a finite time with the swarms of fat robots.*

My gathering solution works synchronously and the robots are represented as closed discs and they are not transparent (also called fat robots). Due to practical considerations, the robots have a limited visibility, they do not have global navigation system, the robots are oblivious (does not have memory) and they are unidentifiable. My goal was to create a reliable gathering algorithm with the above mentioned settings.

I assume that the visibility graph is connected at the beginning which is a primary condition of the successful gathering of robots with limited visibility. The visibility graph must be connected during the gathering otherwise more nodes are created and the gathering will be unsuccessful. The fat robot representation involves some problems such as the robots can block each other's movements and a robot could hide another robot from the view angle of the observer. The basic idea of my solution is the robots move towards the furthest visible robot, which method has not been applied before and which partly provides a solution to the blocking problem. In my solution, the set of robots are divided into interior and perimeter robots subsets. I define a robot as a perimeter robot that sees neighbouring robots at most in 120° (the perimeter robot located at the border of the swarm) the others are denoted as interior robots. This partitioning is executed in every synchronous cycle.

The main aim of my gathering method is the following: the perimeter robots have to move towards the interior robots as fast as they can. The outcome of using only this approach alone is that the visibility graph is disintegrated and more nodes are established. However, this can be prevented

with slowing down the interior robot movements and using the movement limitation algorithm [7].

One of the disadvantages of the fat robot representation is that swarm members could interfere with other robot movements, thus the robots could go into a deadlock state. In order to dissolve this situation a simple method is used: the blocked robot should modify its original goal vector so that the new direction is tangential to the blocker robot, thus the robot can get closer to its originally planned position and the blocked robot can move away from the blocker as well. This method is only applied when the number of blockers is one since more blockers could mean the robot has reached its final destination.

In order to test my algorithm a MATLAB simulation has been performed. My method is compared with an algorithm from the literature an original ([7]) a modified version, which has similar settings like my gathering solution. I measured the necessary execution time of the algorithm and the diameter of the final swarm shape. It can be seen from the results that the necessary gathering time was similar in all three solutions, however, the final shape was smaller in diameter in my method than in the two other cases. According to the experiments, my solution gathered the robots into more compact shapes.

2.2. Synchronous gathering algorithms in asynchronous cases

Thesis 5: *Ando, Suzuki and Yamashita's synchronously working and point robot representation algorithm will successfully fulfil the gathering in asynchronous settings with fat represented robots as long as the swarm member detection time converges to zero.*

In case of asynchronous setting the basic operating conditions are changed in contradiction with the synchronous setting, where the lifetime activity tasks ($Look$ – $Calculate$ – $Move$) start at the same time for every robot. The main problem is swarm member detection, corresponding execution time and the computational performance of robots. Therefore I assume if necessary calculation time of $Look$ and $Calculate$ is near to zero,

the rate of failed gatherings will be reduced. My aim was to examine the rate of visibility graph splitting in different size robot swarms. In this case the minimal robot features are used during the algorithm executions.

During the simulations the previously introduced swarm member detection and distance estimation algorithm is used. With the help of this method the robots are able to determine the relative position of the surrounding robots with a camera sensor. I assumed that the robots are able to capture omnidirectional image and the swarm members are equipped with the periodic robot pattern. Before the simulation the execution time of *Look* phase is determined with the use of implemented algorithm of swarm member recognition. The other parameters such as visibility limit, robot radius, maximal travelling speed are from the swarm member recognition algorithm and from the Surveyor SRV-1 robot.

Before the simulations, the initial positions had been generated, where the initial visibility graphs were connected. The initial positions of the robots were generated in a square area by a uniform random generator with the spatial density of $1/9$ robot$/m^2$. Using this density value resulted in an average robot-robot distance close to the visibility radius (V). After this generation, the largest connected visibility graph was chosen to perform the gathering experiment. Due to this method, the number of the robots in the swarm (N) is not determined a priori, but a random value in each experiment. In order to obtain the effect of N on the splitting probability we defined 9 intervals of N with equal width of 5. These are the following: [5..9], [10..14], [15..19], [20..24], [25..29],[30..34], [35..39], [40..44], [45..49]. In each initial position case the same density was used and 100 different initial visibility graphs for each interval of N was generated, thus altogether 900 visibility graphs became available.

During the simulation I used the synchronous gathering algorithm by Ando [7] with asynchronous settings since in [8] the authors introduced a special case when the visibility graph is split. Altogether 3600 experiments were performed with my MATLAB simulation program with the above described initial positions and for four different computational performances of the robots hardware. To sum up, 900 experiments were conducted with the 900 initial positions for each specific computational performance. I

determined the frequency of splitting of the visibility graph, and I obtained an estimation of the probability of swarm splitting regarding the number of robots and for lower and faster computational performance.

I denoted the computational performance of Surveyor SRV-1 robot by *T1*, which is the time necessary for the robot to be able to detect the neighbouring robots with the above introduced swarm member recognition algorithm. I introduced three other (faster) computational performances, which will be denoted by *T2*, *T3* and *T4*. *T2* means 5 times faster performance and *T3*, *T4* mean 10 times and 20 times better computational performance, where I preliminary expected the number of visibility graph splitting decrease along with decrease of the necessary calculation time to 0. Note that during the simulations, the maximum travelling speed of robot is applied, but I have had the same results with 1/5, 1/10 and 1/20 times slower travelling speed.

I determined two stopping conditions: the first when the swarm is gathered (a contact graph is connected); and the other when the visibility graph is split. All units in the swarm repeat the $Look - Calculate - Move$ cycle until one of the stopping conditions are satisfied.

From the results it can be clearly seen that asynchronous gathering is possible with fat robots if the necessary calculation time of life activities (*Look* and *Calculate*) converge to zero and a suitable travelling speed is used. In case of point robot representation starting from the same initial robot positions the swarm significantly split on several occasions, which leads to the conclusion that in case of fat robot representation the robots are not able to leave each other because of the collisions. By all means, the SEC based gathering algorithm [7] has an effect on the results, which moves the robots towards the centre of the smallest enclosing circle of visible robots.

2.3. Publications

In connection with the gathering algorithms 3 book chapters ([9] [10] [11]) have been published, 2 of them have DOI identifier and are indexed by the DBLP. I have 5 independent references to [9] article and 3 for [10]. Moreo-

ver, I have already published the results in a journal [12] and a conference article ([13]) in Hungarian.

3. Helping indoor navigation with visual information

Thesis 6: *I have developed a visual perception system based indoor navigation helper method for mobile robots, which approximates a well-calibrated, odometry-based navigation accuracy.*

In my method the relative position of the robot is determined by the feature objects placed in the environment. My main goal was to develop a method, which is as accurate as a well-calibrated odometry-based method, but it can be used more widely. In case of the odometry method, the robots and the sensors are calibrated individually. However, in my method the calibration is executed only once and all the robots, which have the same settings, could use this method with different parameters.

In order to localise and to explore the working environment, the robot only uses its own camera system to navigate. I assume that the robot works in an indoor environment, which is a flat ground and the ground is covered with a pattern. This pattern is a priori known for the robot. The camera, which is equipped in a fixed position, captures images from the local environment and the pattern is clearly visible for the sensor. With the help of the camera, the robot observes the environment and the captured images are processed and the featured objects are enhanced by segmentation methods. Since the object properties are a priori known it is possible to obtain some geometrical information. In order to get the displacement and the relative position of the robot, the enhanced objects are labelled and tracked in the future frames. Moreover, the robot is able to explore the environment that is based on the floor pattern.

The first step of my image pre-processing method is grayscale conversion, after that the useful pixels are selected by an adaptive thresholding method. I assume that the foreground and the background objects are

separable, due to their intensity values. The reason why an adaptive thresholding method must be used is that the lighting conditions and the shadows of the robot make thresholding difficult and in this case only one global threshold value is not suitable. The thresholded image contains some salt and pepper types of noise, which are suppressed with a median filter. The next step is the calculation of the perspective transformation and the determination of the undistorted top-view image, which allows for detecting parallel and perpendicular features on the result image. On the pre-processed image a Hough-line detection algorithm is executed and the line information is stored in the polar coordinates. Hough transformation was used with a low threshold value, thus numerous line candidates were selected by the algorithm. Due to this, the line candidates were clustered and after this, the best fitting lines were selected from each cluster.

The whole pattern is detected by the line objects. In the first step, the parallel line pairs are determined and from these pairs the perpendicular line pairs are also classified. At this stage floor pattern properties are also taken into consideration and the selected line pairs should not be too close to or too far from each other. With this method the false positive line detections are discarded since after using this method only the pattern-forming line objects remain.

In order to track the selected objects the primitives of the pattern have to be labelled, and their further identification has to be solved as well. The tracking of the objects is based on the distance metric, where the distance is calculated from the current and the previously captured frame. The robots store the numbers of incomplete identifications and the number of successful identifications as well.

While running the agent program, the robot stores the displacement vector between the two captured frames in order to build the travelled trajectory. In my solution the identified object points are ordered descending by the numbers of detection in previous images. In the first sampled frame the vector calculation is based on all of the object points, but in the further calculations only those object points are used that have been detected more times. In each sampling the average of displacement is calculated and stored. The sequence of these vectors represents the trajectory.

This proposed algorithm has been tested with numerous experiments. My main goal was to examine the accuracy of the suggested method and analyse the value of cumulative errors after the robot covered a greater distance. The accuracy of my method was compared with odometry-based methods [14] [15] [16] where the relative error was 2.0E-3. The relative error is defined by $\Delta L/L$, where ΔL means the error of relative position after the robot takes an L length path [17]. In case of odometry, the relative error is constant, thus the uncertainty increases with the covered distance.

In the experiments the robot took a 20 m length straight path. During its travelling I stored the camera frame index after every 2.5 m, thus in the 20 m path 8 milestones are registered, which is important for the accuracy of the measurement. Altogether I captured 15 experiments, which were processed afterwards with my adapter software. The available 15 experiments were divided into 10 calibration and 5 test parts. The results of these experiments show that the relative error was higher than with the odometry-based method. However, at first the relative error came close to the odometry error limit and after that it went below the limit of 2.0E-3.

3.1. Publications

The proposed indoor navigation helper method is published in conference proceedings [18].

Irodalom és hivatkozások – Bibliography and references

[1] K. Bolla, T. Kovács, and G. Fazekas. A fast visual perception system based kin recognition and distance evaluation method in a robot swarm. *GAMF Közlemények*, pages 49–61, 2010.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] K. Bolla, T. Kovács, and G. Fazekas. Distinguish competitor robot swarms with on-board visual perception system. *CSCS PhD Conference*, 2010.

[7] 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.

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

[9] 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.

[10] 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.

[11] 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.

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

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

[14] 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.

[15] 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.

[16] 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.

[17] 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.

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