16

Navigation for mobile autonomous robots and their formations: An application of spatial reasoning induced from rough mereological geometry

Lech Polkowski and Pawel Osmialowski
Polish - Japanese Institute of Information Technology
Poland

1. Introduction

This Chapter is intended as a sequel to our Chapter 21 in the book on "Mobile Robots Motion Planning. New Challenges" by this Publisher. It is now commonly accepted that problems of planning and navigation are inseparable: "Most recent contribution to the field combine effective algorithms tested on significant problems, along with some formal guarantees of performance" (J.-C. Latombe in Foreword to "Principles of Robot Motion. Theory, Algorithms and Implementations" by H. Choset et al.). Therefore, as with the former Chapter, we provide in this Chapter planning mechanisms along with navigation tests and a theoretical analysis of underlying constructs. We extend our scope of analysis by considering formations of mobile autonomous robots. We introduce a definition of a robot formation, based on the spatial relation of betweenness and we give a treatment of planning and navigation problems for robot formations. In our investigations into problems of multi-robot planning and navigation, we apply rough mereological theory of spatial reasoning. This theory is briefly recalled in this Chapter for completeness sake. The software system Player/Stage is employed as means of simulation and visualization of robot trajectories to chosen goals. To this end, it has been provided with SQL functions rendering predicates of rough mereological geometry. Robotics of autonomous mobile robots presents the most intricate field for applications of techniques of artificial intelligence, decision making and cognitive methods. Among the basic problems in this area are planning and navigation problems and we are concerned with them both in their mutual bond.The planning and navigation problem for mobile robots is addressed from many angles and a multitude of approaches and techniques have emerged; it suffices to mention a division of planners according to assumptions about robot equipment and abilities as well theoretical principles used in planner construction, from simple bug-type algorithms though potential functions and potential field based strategies to roadmaps constructed by exploiting visibility in configuration spaces, metric-based ideas like Voronoi diagrams and graphs, cell decompositions of various types, and probabilistic (sampling) planners allowing for incremental space exploration by building trees of configuration points like EST-trees or

RRT-trees, see (Choset et al., 2005) for an excellent account of these approaches.Path planning methods, according to (Latombe, 1991) can be divided into *centralized*, in which case planning considers all robots in a team, or *decoupled*, when path is planned for each robot independently. Another division of path planning methods consists in *local* vs. *global* approach; in the local method, planning is provided for some neighborhood of a robot, whereas in the global approach, the whole environment is taken into consideration. Path planning consists in finding controls which secure the desired path of a robot toward a goal with, e.g., obstacle avoidance. As with a single robot, the path planning problem arises for teams of robots. In particular, centralized methods for single robots are extended to teams of robots see, e.g., (Švestka&Overmars, 1998) where such planning is applied with help of relational structures called super-graphs on which admissible paths are searched for. This approach however assumes that the situation is static, i.e., no changes in the environment happen during plan execution. The assumption of environment stability is not valid in real world situations and the reactive approach (Arkin, 1998) to planning considers simple, sensor - actuator coupling schemes expressible as low-level behaviors (Urdiales et al., 2006) ; in these schemes, potential field methods, vector field histograms, dynamic window approach are used (Urdiales et al., 2006). Some reactive methods use dynamic variants of search algorithms like A*, e.g., D* (Brumitt et al., 2001).

From among those methods, we choose to adopt the method of potential field, see sect.5. In classical setting, the potential field is built as the sum of two components: repulsive, induced by obstacles, and attractive, induced by goals. The field force is defined as the gradient of the repulsive, respectively, attractive, potential, see (Choset et al., 2005). In either case, the potential is defined with the use of a metric, in analogy to classical physical examples of a potential field like Coulomb or gravitational fields. Our approach is different: the potential field is constructed by means of a chosen rough inclusion - the primitive predicate of rough mereology, see sect.5. A robot is driven to the goal by following areas of increasing density of the field as shown in sect.5. The problem for a single robot is presented fully in (Polkowski&Osmialowski, 2009) where mereological potential fields have been constructed and applied in planning of paths and robot navigation.

Problems of cooperative mobile robotics are even more demanding as they require an accounting for group behavior of many autonomous mobile robots. There is the increasing need for making use of such teams in practical problems of performing complex tasks inaccessible for a single robot (like pushing large objects, rescue operations, assembling); there is also a theoretical interest in research on aspects of their behavior: cooperative mechanisms, leadership, conflict resolution, consensus making, many of which belong as well in biology and environmental studies, see, e.g., (Balch&Arkin, 1998; Brumitt et al., 2001; Chen&Luh, 1998; Leonard&Fiorelli, 2001; Shao et al., 2005) and also a discussion in (Cao et al., 1997). These motifs have propelled research in direction of multi-robot planning.

Cooperative behavior is perceived by many authors, see, e.g., (Cao et al., 2005) and references therein, as a specialization of collective behavior having the tint of achieving jointly some goal. The goal may mean an economic advantage, reaching a specified position, learning jointly a feature or a category of objects, etc., etc. Main directions of research in this area of schemes for cooperative mobile robotics, include, as distinguished in the literature,

see, e.g., (Cao et al., 2005; Kramer&Scheutz, 2007), a study on group architectures, conflicts of resources, motivations for cooperation, learning of a cooperative behavior, spatiotemporal aspects: path planning, moving to formations, pattern generation.

In this work, which extends our earlier results (Osmialowski, 2009; Polkowski&Osmialowski, 2008a), we are concerned with the last aspect, i.e., moving to formations and path planning in order to make robots into a given formation. Alongside, we are concerned with problems of repairing formations and navigating formations in static environments. We study the problem of path planning in order to make robots in a team into a formation. We apply as a theoretical framework for our approach, a qualitative theory of spatial reasoning as provided by Rough Mereology, see, e.g., (Polkowski&Osmialowski, 2008). In this framework, we give a definition of a formation by a team of robots, by means of a rough mereological betweenness relation among them, see (Osmialowski, 2009; Polkowski & Osmialowski, 2008). In the same framework, we study the problem of path planning for moving into a given formation. We propose some procedures to this end. We model our robots on Roomba[1] robots by iRobot(R), i.e., we assume our robots to be planar disk-shaped objects. We use Player/Stage system, see (Osmialowski, 2007; 2009; http://playerstage.sourceforge.net) , as a tool for simulation and visualization.

2. On formations of autonomous mobile robots

A study of the concept of a robot formation was initially based on a perception of animal behavior like herding, swarming, flocking or schooling. In this respect, a few principles emerged, see, e.g., (Balch&Arkin, 1998), keeping all animals within a certain distance from one another (e.g., to ensure mutual visibility), moving away when the distance becomes too close (to avoid congestion, collision, or resource conflict), adapting own movement to movement of neighbors (e.g., velocity of motion), orienting oneself on a leader.

From those observations a geometric approach to formations has been derived: a formally simplest approach (Balch&Arkin, 1998), uses referencing techniques; reference is made either to the team center or to the team leader, or to a specified neighbor in a coordinate system given by the position of the team center or the leader along with the orientation given by the nearest navigation point; positions are determined, e.g., with the help of GPS or dead reckoning. Another method for forming a geometric formation relies on a direct usage of a metric, say rho, see, e.g., (Chen&Luh,1998; Sugihara&Suzuki,1990): given a threshold δ, and a parameter ε, for each robot r in a team, its farthest neighbor $r1$ and the nearest neighbor $r2$, if $\rho(r, r1) > \delta$ then r moves toward $r1$, if $\rho(r, r1) < \delta - \varepsilon$ then r moves away from $r1$, if $\delta - \varepsilon < \rho(r, r1) < \delta$ then r moves away from $r2$. By this method, robots are arranged on a circle. Some methods rely on the potential field technique (Leonard& Fiorelli, 2001); in this approach, the potential of the field is defined dependent on the distance among robots in the team in order to keep distances among them as prescribed. In addition, also the technique of a virtual leader is involved to keep robots in a team at a prescribed distance from their current leaders; in some approaches the relation the leader - the follower is expressed by means of control laws in a given coordinate system (Das et al., 2002; Shao et al., 2005), with execution of movement controlled by an omnidirectional camera.

1 Roomba is the trademark of iRobot Inc.

It seems desirable to propose an approach which in principle would be metric independent and which would take into account only relative positions of robots one to another. In this work we propose a definition of a formation which is based on spatial predicates defined within rough mereological theory of spatial reasoning, see, e.g., (Polkowski, 2001; Polkowski&Osmialowski, 2008; 2008a).

3. Qualitative spatial reasoning: a nutshell reminder

In this Section, we recall elements of spatial theory induced in the rough mereological framework which have already been presented extensively elsewhere, in particular in (Polkowski&Osmialowski, 2008), Ch. 21. Qualitative Spatial Reasoning emerged on basis of an idea by (A. N. Whitehead, 1929; Leonard & Goodman, 1940) of an extension, dual to a notion of part in mereology theory (Lesniewski,1916;1982), reformulated as a theory of Connection (Leonard&Goodman, 1940; Clarke, 1981). Qualitative Spatial Reasoning is a basic ingredient in a variety of problems in mobile robotics, see, e.g., (Kuipers&Byun, 1987). Spatial reasoning which deals with objects like solids, regions etc., by necessity refers to and relies on mereological theories of concepts based on the opposition part - whole (Gotts et al., 1996). Mereological ideas have been early applied toward axiomatization of geometry of solids, see (De Laguna, 1922; Tarski,1929).

Mereological theories rely either on the notion of a part (Lesniewski, 1916; 1982), or on the notion of objects being connected (Clarke, 1981; Gotts et al., 1996). Our approach to spatial reasoning is developed within the paradigm of rough mereology. Rough mereology, see, e.g., (Polkowski, 2003; 2004; 2008), is based on the predicate of being a part to a degree, called a *rough inclusion* and thus it is a natural extension of mereology based on part relation, as proposed by (Lesniewski, 1916; 1982). A rough inclusion, cf., (Polkowski, 2008), is a ternary relation μ such that for any pair of objects u, v and real r the formula $\mu(u, v, r)$ means that *u is a part of v to a degree of r* where $r \in [0,1]$.

In our applications to spatial reasoning, objects will be regions in Euclidean spaces, notably rectangles, in particular squares, or discs in 2-dimensional space, and the rough inclusion applied will predominantly be the one defined by the equation,

$$\mu^0(u, v, r) \text{ if and only if } \frac{|u \cap v|}{|u|} \geq r \quad (1)$$

where $|u|$ is the area of the region u.

On the basis of a given rough inclusion μ, we can introduce predicates of a certain geometry of regions in low-dimensional spaces. Points in such geometries are recovered usually be means of the technique proposed by Alfred Tarski of ultrafilters of regions, see (Tarski, 1929).

4. Mereogeometry: a geometry of regions

We are interested in introducing into the mereological world defined by μ^0 a geometry in whose terms it will be possible to express spatial relations among objects; a usage for this geometry will be found in navigation and control tasks of multi-agent mobile robotics.

4.1 A notion of a quasi-distance

We first introduce a notion of a quasi-distance κ in our rough mereological universe by letting,

$$\kappa(u, v) = \min\{\text{argmax}_r \mu^0(u, v, r), \text{argmax}_s \mu^0(v, u, s)\} \quad (2)$$

Observe that mereological distance differs essentially from the standard distance: the closer are objects, the greater is the value of κ : $\kappa(u, v) = 1$ means $u = v$, whereas $\kappa(u, v) = 0$ means disjointness in the sense of μ^0 of u and v regardless of the Euclidean distance between them.

4.2 Nearness and Betweenness: Van Benthem's variant

We apply the distance κ to define in our context the predicate N of nearness proposed in (van Benthem, 1983),

$$N(z, u, v) \Leftrightarrow (\kappa(z, u) = r, \kappa(u, v) = s \Rightarrow s < r) \quad (3)$$

Here, nearness means that z is closer to u than v is to u.

We make an essential use of the betweenness predicate T_B proposed by van Benthem [3], in analogy to the Tarski betweenness (Tarski,1959) on the basis of the nearness predicate,

$$T_B(z, u, v) \Leftrightarrow [\text{for all } w \ (z \text{ is } w \text{ or } N(z, u, w) \text{ or } N(z, v, w))] \quad (4)$$

Example 1. We consider a context in which objects are rectangles positioned regularly, i.e., having edges parallel to axes in R^2. The measure μ is μ^0 of (1). In this setting, given two disjoint rectangles C, D, the only object between C and D in the sense of the predicate TB is the extent $ext(C, D)$ of C, D, i.e., the minimal rectangle containing the union $C \cup D$. As linear stretching or contracting along an axis does not change the area relations, it is sufficient to consider two unit squares A, B of which A has (0, 0) as one of vertices whereas B has (a, b) with $a, b > 1$ as the lower left vertex (both squares are regularly positioned). Then the distance κ between the extent $ext(A, B)$ and either of A, B is $\frac{1}{(a+1)(b+1)}$. For a rectangle R: [0, x] x [0, y] with $x \in (a, a+1)$, $y \in (b, b+1)$, we have that $\kappa(R, A) = \frac{(x-a)(y-b)}{xy} = \kappa(R, B)$. For $\phi(x, y) = \frac{(x-a)(y-b)}{xy}$, we find that

$\frac{\partial \phi}{\partial x} = \frac{a}{x^2} \cdot (1 - \frac{b}{y}) > 0$, and, similarly, $\frac{\partial \phi}{\partial y} > 0$, i.e., *ø* is increasing in *x*, *y* reaching the maximum when *R* becomes the extent of *A*, *B*. An analogous reasoning takes care of the case when *R* has some (*c*, *d*) with *c*, *d* > 0 as the lower left vertex.

The betweenness predicate allows for definitions of various *patterns Pt*. For instance, we define a *line pattern*. We let,

$$Pt(u, v, z) \text{ <=> } z \text{ is } TB(u, v) \text{ or } u \text{ is } TB(z, v) \text{ or } v \text{ is } TB(u, z) \quad (5)$$

We will say that a finite sequence *u1*, *u2*, ..., *un* of objects belong in a line segment whenever *Pt*(*ui*, *ui*+*1*, *ui*+*2*) for *i* = 1, ..., n - 2; formally, we introduce the functor *Line* of finite arity defined via

$$Line(u1, u2, ..., un) \text{ <=> for all } i < n - 1 : Pt(ui, ui+1, ui+2) \quad (6)$$

Example 2. With reference to Example 1, rectangles *C*, *D* and their extent *ext*(*C*, *D*) form a line segment.

5. Mereological potential fields

As mentioned in sect.2, the technique of potential fields, see (Krogh, 1984; Khatib, 1986) for seminal ideas, cf., (Choset et al., 2005; Latombe, 1991), well-known from planning in case of a single robot, has been extended to the case of robot teams. An example of this approach is given in (Leonard&Fiorelli, 2001), where robots in a team are organized around a set of beacons called *leaders*, and are subjected to repulsive and attractive forces induced by potential fields generated for pairs of robots and pairs of the form robot–leader in such a way as to prevent too close distance among robots and to keep them along leaders.

In our case, we apply the idea already exposed, see (Osmialowski, 2009; Osmialowski& Polkowski,2009; Polkowski&Osmialowski, 2008a), of building a potential field from the rough inclusion μ^0. Our path planner accepts target point coordinates and provides a list of *waypoints* from a given robot position to the goal. It takes as an input a map of static obstacles that a robot should avoid while approaching the target point. A robot and a target should both lay within the area delimited by surrounding static obstacles that form borders of the robot environment. There can be other static obstacles within the area, all marked on the provided map. After the path is proposed, a robot is lead through the path until it reaches given target. If a robot cannot move towards the target position for some longer time (e.g., it keeps on hitting an other robot reaching its target or some unknown non-static obstacle), a new path is proposed.We tested our planner by running simulations in which we have had a model of Roomba robot, see (Tribelhorn&Dodds, 2007) traveling inside an artificially created environment. Real Roomba robots are disc-shaped and therefore easy to model, but they do not provide many useful sensor devices (except bumpers which we were using to implement lower-level reaction to hitting unexpected obstacles). Also, odometry of

Roomba robots is unreliable (loc.cit) hence we assume that simulated robots are equipped with a global positioning system.Right after the target position is given, our planner builds the mereological potential field filled with squared areas each of the same size. The field is delimited by environment's borders. Only space free of obstacles is filled.
The algorithm for building the potential field is the following.

SQUARE_FILL_ALGORITHM
Structure: a queue Q
1. Add to the queue Q, x and y coordinates of a given goal together with 0 as current distance from current squared area to the next neighboring area (so they will be part of each other to the maximal degree). Also put clockwise as current direction of exploration. These are initial values.
2. Spin in the main loop until there are no more elements in the queue Q:
2.1. Extract x, y, current distance and current direction of exploration from the beginning of queue Q.
2.2. Check if there is any other squared area already present in potential field to which the distance from current x and y coordinates is equal or shorter than current distance. If so, skip taken element and run new main loop turn.
2.3. Form new squared area with current x and y as the coordinates of the centroid of this new area. Check if there are any common part with any static obstacle within this new squared area. If so, skip taken element and run new main loop turn.
2.4. Add new squared area to the potential field.
2.5. Increase current distance by 0.01.
2.6. Add eight neighbor areas to the queue Q (for each area add these data: x and y coordinates, current distance and direction of exploration opposite to current); if direction is clockwise neighbors are: left, left-up, up, right-up, right, right-down, down, left-down; if direction is anti-clockwise neighbors are: left-down, down, right-down, right, right-up, up, left-up, left.
2.7. Run new main loop turn.

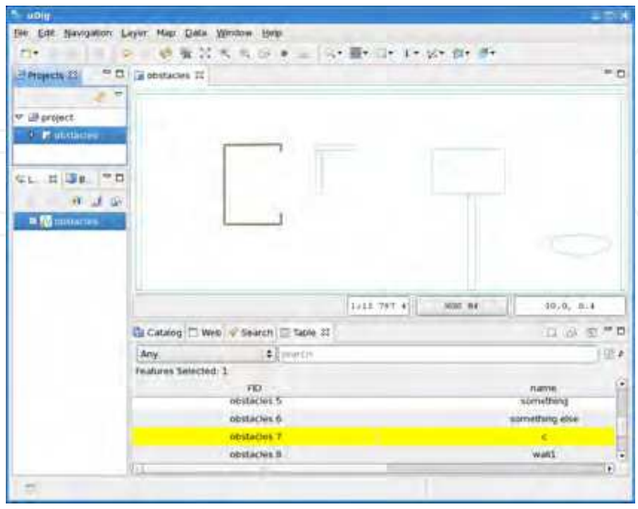

Fig. 1. Map of our artificial world edited by the *uDig* application (created and maintained by Refractions Research). The map consists of number of layers whose can be edited individually; on the figure we can see how solid obstacles are situated within *obstacles* layer

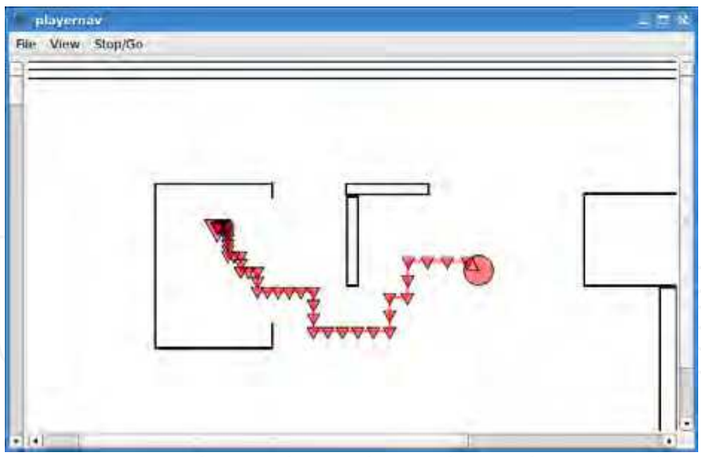

Fig. 2. The *playernav* program can be used to indicate the goal position for given robot and to show trajectory computed by underlying planer. In his situation, our mereological planner computed the trajectory.

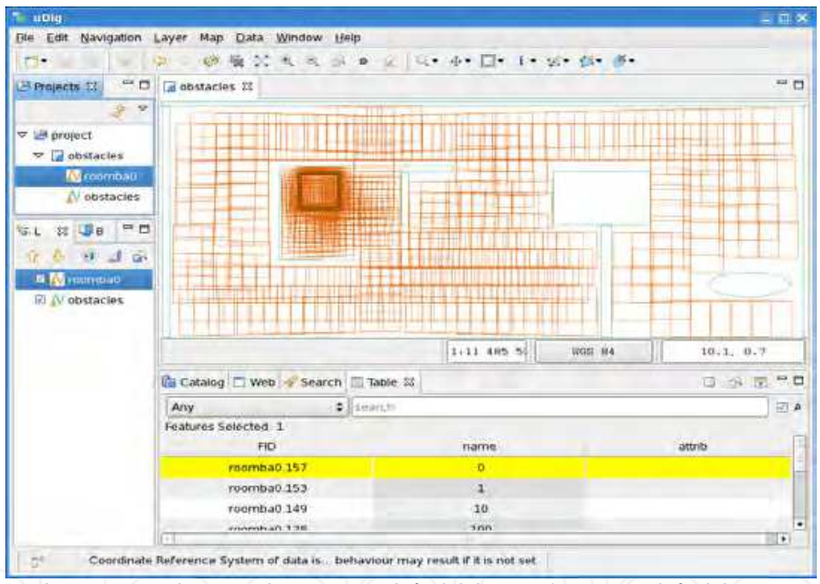

Fig. 3. *Obstacles* layer together with potential field layer (potential field generated for given goal is stored as another map layer, here called *roomba0*). Observe increasing density towards the goal.

6. A definition of a formation of robots

We propose a theory of many robot structures, based on the predicates of rough mereological geometry, of which foremost is the predicate T_B of betweenness. A Roomba robot is a disc-shaped robot and due to this we model it as the square circumscribing the robot with edges parallel to coordinate axes of the reference system. This allows for the extent of two given robots to be always oriented as a regular rectangle, i.e., with edges parallel to coordinate axes. In particular, this feature allows for translational and rotational invariance of extents, more generally under affine transformations of the plane.

Definition 1. *We say that a robot B is between robots A and C, in symbols* (*between B A C*), *in case the rectangle ext*(*B*) *is contained in the extent of rectangles ext*(*A*), *ext*(*C*), *i.e.*, μ^0(*ext*(*B*), *ext*(*ext*(*A*), *ext*(*C*)), 1).

This allows as well for a generalization to the notion of *partial betweenness* which models in a more precise manner spatial relations among *A, B, C* (we say in this case that robot *B* is between robots *A* and *C* to a degree of at least *r*): in symbols,

$$(\text{between-deg } r\ B\ A\ C) \tag{7}$$

if and only if

$$\mu^0(ext(B), ext[ext(A), ext(C)], r) \tag{8}$$

We now give the central definition in this work: the definition of a formation. By a formation, we mean a set of robots along with a structure imposed on it as a set of spatial relations among robots.

Definition 2. *For a team of robots, T*(*r*1, *r*2, …, *rn*) = {*r*1, *r*2; …, *rn*}, *an ideal formation IF on T*(*r*1, *r*2, …, *rn*) *is a betweenness relation (between …) on the set T*(*r*1, *r*2, …, *rn*) *of robots.*

In practice, ideal formations will be given as a list of expressions of the form,

$$(\text{between } r0\ r1\ r2) \tag{9}$$

indicating that the object *r*0 is between *r*1; *r*2, for all such triples, along with a list of expressions of the form,

$$(\text{not-between } r0\ r1\ r2) \tag{10}$$

indicating triples which are not in the given betweenness relation.

To account for dynamic nature of the real world, in which due to sensory perception inadequacies, dynamic nature of the environment, etc., etc., we allow for some deviations from ideal formations by allowing that the robot which is between two neighbors can be between them to a degree in the sense of (7).

This leads to the notion of a real formation.

Definition 3. *For a team of robots, T*(*r*1, *r*2, …, *rn*) = {*r*1, *r*2, …, *rn*}, *a real formation RF on T*(*r*1, *r*2, …, *rn*) *is a betweenness to degree relation* (*between-deg …*) *on the set T*(*r*1, *r*2, …, *rn*) *of robots.*

In practice, real formations will be given as a list of expressions of the form,

$$(\text{between-deg } \delta\ r0\ r1\ r2) \tag{11}$$

indicating that the object $r0$ is to degree of δ in the extent of $r1$, $r2$, for all triples in the relation (between-deg …), along with a list of expressions of the form,

$$\text{(not-between } r0\ r1\ r2\text{)} \tag{12}$$

indicating triples which are not in the given betweenness relation.

In Fig. 4, we sketch some cases of instances of relations (between-deg δ $r0$ $r1$ $r2$).

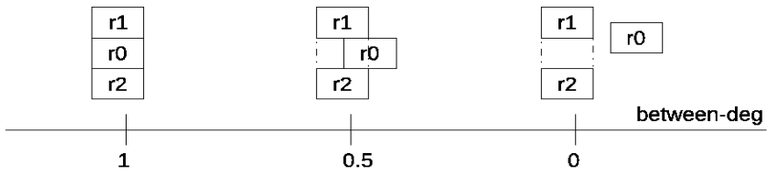

Fig. 4. Object r_0 is in extent of r_1 and r_2 to degree δ.

7. On complexity of formation description

Description of formations, as proposed in Def. 1, 2 of sect. 6, can be a list of relation instances of large cardinality, cf., Examples 3 and 4, below. The problem can be posed of finding a minimal set of instances wholly describing a given formation. It turns out (Polkowski&Osmialowski,2008) that the problem is intractable. We have

Proposition 1. *The problem of finding a minimum size description of a given formation is NP-hard.*

8. Implementation in Player/Stage software system

Player/Stage is an Open-Source software system designed for many UNIX-compatible platforms, widely used in robotics laboratories (Kramer&Scheutz, 2007; Osmialowski, 2007; http://playerstage.sourceforge.net) . Main two parts are Player - message passing server (with bunch of drivers for many robotics devices, extendable by plug-ins) and Stage - a plug{in for Player's bunch of drivers which simulates existence of real robotics devices that operate in the simulated 2D world.

Player/Stage offers client-server architecture. Many clients can connect to one Player server, where clients are programs (robot controllers) written by a user who connects to Player client-side API. Player itself uses drivers to communicate with devices and in this activity it does not make distinction between real and simulated hardware. It gives the user means for testing programmed robot controller in both real and simulated world.

Among all Player drivers that communicate with devices (real or simulated), there are drivers not intended for controlling hardware, instead those drivers offer many facilities for sensor data manipulation, for example, camera image compression, retro-reflective

detection of cylindrical markers in laser scans, path planning. One of the new features added to Player version 2.1 is the PostGIS[2] driver: it connects to PostgreSQL database in order to obtain and/or update stored vector map layers.

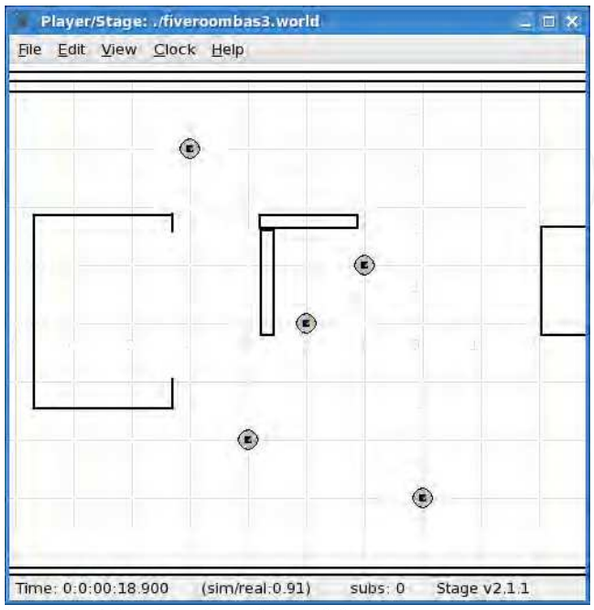

Fig. 5. Five Roomba robots inside simulated world

PostGIS itself is an extension to the PostgreSQL object-relational database system which allows GIS (Geographics Information Systems) objects to be stored in the database. It also offers new SQL functions for spatial reasoning. Maps which are to be stored in SQL database can be created and edited by graphical tools like *uDig* or by C/C++ programs written using GEOS library of GIS functions. PostGIS, uDig and GEOS library are projects maintained by Refractions Research. A map can have many named layers, and for each layer a table in SQL database is created. We can assume that the layer named *obstacles* consists of objects which a robot cannot walk through. Other layers can be created in which we can divide robot's workspace into areas with an assigned attribute which for example tells whether a given area is occupied by an obstacle or not. During our experimentations, we have created a plug-in for Players bunch of drivers which constantly tracks changes of position of every robot and updates *obstacles* layer so robots are marked as obstacles. As a result, the map stored in SQL database is kept always up to date. This feature is also useful in multi-agent environments: at any time a robot controller can send a query to SQL database server regarding every other robot position.

8.1. SQL queries representing rough mereogeometric predicates

A roboticist can write a robot controller using Player client-side API which obtains information about current situation through the *vectormap* interface. Additionally, to write such a program, PostgreSQL client-side API can be used in order to open direct connection to the database server on which our mereogeometry SQL functions are stored together with

[2] PostGis is an intelectual property of Refractions Research, Inc.

map database. These functions can be called using this connection and results are sent back to the calling program. This gives robot controller program ability to perform spatial reasoning based on rough mereology. Using PostGIS SQL extensions we have created our mereogeometry SQL functions, see (Ladanyi, 1997). Rough mereological distance is defined with help of the following SQL function *meredist*.

```
CREATE FUNCTION meredist(object1 geometry, object2 geometry)
RETURNS DOUBLE PRECISION AS
$$
        SELECT min(degrees.degree) FROM
                ((SELECT
                        ST Area(ST Intersection(extent($1), extent($2)))
                        / ST Area(extent($1))
                        AS degree)
                UNION (SELECT
                        ST Area(ST Intersection(extent($1), extent($2)))
                        / ST Area(extent($2))
                        AS degree))
                AS degrees;
$$ LANGUAGE SQL STABLE;
```

Having mereological distance function we can derive nearness predicate:

```
CREATE FUNCTION merenear(obj geometry, o1 geometry, o2 geometry)
RETURNS BOOLEAN AS
$$
        SELECT meredist($1, $2) > meredist($3, $2)
$$ LANGUAGE SQL STABLE;
```

The equi-distance can be derived as such:

```
CREATE FUNCTION mereequ(obj geometry, o1 geometry, o2 geometry)
RETURNS BOOLEAN AS
$$
        SELECT (NOT merenear($1, $2, $3))
                AND (NOT merenear($1, $3, $2));
$$ LANGUAGE SQL STABLE;
```

Our implementation of the betweenness predicate makes use of a function that produces an object which is an extent of given two objects:

```
CREATE FUNCTION mereextent(object1 geometry, object2 geometry)
RETURNS geometry AS
$$
        SELECT GeomFromWKB(AsBinary(extent(objects.geom))) FROM
                ((SELECT $1 AS geom)
```

```
                UNION (SELECT $2 AS geom))
                AS objects;
$$ LANGUAGE SQL STABLE;
```

The betweenness predicate is defined as follows:

```
CREATE FUNCTION merebetb(obj geometry, o1 geometry, o2 geometry)
RETURNS BOOLEAN AS
$$
        SELECT
                meredist($1, $2) = 1
                OR meredist($1, $3) = 1
                OR
                        (meredist($1, $2) > 0
                        AND meredist($1, $3) > 0
                        AND meredist(mereextent($2, $3),
                                mereextent(mereextent($1, $2), $3)) = 1);
$$ LANGUAGE SQL STABLE;
```

Using the betweenness predicate we can check if three objects form a pattern:

```
CREATE FUNCTION merepattern(object1 geometry, object2 geometry, object3 geometry)
RETURNS BOOLEAN AS
$$
        SELECT merebetb($3, $2, $1)
                OR merebetb($1, $3, $2)
                OR merebetb($2, $1, $3);
$$ LANGUAGE SQL STABLE;
```

Also having pattern predicate we can check if four objects form a line:

```
CREATE FUNCTION mereisline4(obj1 geometry, obj2 geometry, obj3 geometry, obj4
geometry)
RETURNS BOOLEAN AS
$$
        SELECT merepattern($1, $2, $3) AND merepattern($2, $3, $4);
$$ LANGUAGE SQL STABLE;
```

To figure out if a set of objects form a line an aggregate can be used:

```
CREATE FUNCTION mereisline_state(state array geometry[4], input data geometry)
RETURNS geometry[4] AS
$$
        SELECT ARRAY[$1[2], $1[3], $2, result.object]
                FROM (SELECT CASE
                WHEN $1[4] IS NOT NULL
                        THEN $1[4]
```

```
                WHEN $1[3] IS NULL
                        THEN NULL
                WHEN ($1[2] IS NULL) AND (meredist($1[3], $2) > 0)
                        THEN NULL
                WHEN ($1[2] IS NULL) AND (meredist($1[3], $2) = 0)
                        THEN $2
                WHEN ($1[1] IS NULL) AND merepattern($1[2], $1[3], $2)
                        THEN NULL
                WHEN ($1[1] IS NULL) AND (NOT merepattern($1[2], $1[3], $2))
                        THEN $2
                WHEN merepattern($1[1], $1[2], $1[3]) AND merepattern($1[2], $1[3], $2)
                        THEN NULL
                ELSE $2
        END AS object)
AS result;
$$ LANGUAGE SQL STABLE;

CREATE FUNCTION mereisline_final(state array geometry[4])
RETURNS BOOLEAN AS
$$
        SELECT ($1[4] IS NULL)
                AND ($1[3] IS NOT NULL)
                AND ($1[2] IS NOT NULL);
$$ LANGUAGE SQL STABLE;

CREATE AGGREGATE mereisline
(
SFUNC = mereisline_state,
BASETYPE = geometry,
STYPE = geometry[],
FINALFUNC = mereisline_final,
INITCOND = '{fg}'
);
```

For our convenience we have derived betweenness predicate in more general form:

```
CREATE FUNCTION merebet(object geometry, object1 geometry, object2 geometry)
RETURNS BOOLEAN AS
$$
        SELECT (
                ST Area(ST Intersection(extent($1), mereextent($2, $3)))
                / ST Area(extent($1))
                ) = 1.0;
$$ LANGUAGE SQL STABLE;
```

8.2. A driver for Player server to maintain formations

We have created a plug-in driver (written in C++ programming language) for Player server that keeps on tracking all robots in a team in order to make sure their positions form desired formation. If formation is malformed, our driver tries to repair it by moving robots to their proper positions within the formation. Also our driver is responsible for processing incoming orders: position commands which are dispatched to formation leader (selected member of a team) and geometry queries which are replied with information about current formation extent size and its global position. As such, our driver can be considered as a finite state machine which by default is constantly switching between two states: *process orders* and *formation integrity check*. If formation integrity check fails it switches to *repair formation* state.

Formation integrity check is done according to a given description. As pointed earlier, description of formation is a list of s-expressions (LISP-style symbolic expressions). To parse those descriptions efficiently we have used *sfs-exp* programming library written by Matthew Sottile (sfsexp). Each relation between robots in given description is checked and if related robots positions do not fulfill requirements, error value is incremented. Also while traversing through a description, overall error value is computed in order to figure out what could be the maximum error value for the given description. Finally, error value derived from each robot position is divided by computed overall error value which gives the normalized formation fitness value between 0 (all requirements were fulfilled) and 1 (none of requirements were fulfilled). If the fitness value is below some threshold (typically 0.2), then we can conclude that robots are in their desired positions.

Typical formation description may look like below.

Example 3.

```
(cross
        (set
                (max-dist 0.25 roomba0 (between roomba0 roomba1 roomba2))
                (max-dist 0.25 roomba0 (between roomba0 roomba3 roomba4))
                (not-between roomba1 roomba3 roomba4)
                (not-between roomba2 roomba3 roomba4)
                (not-between roomba3 roomba1 roomba2)
                (not-between roomba4 roomba1 roomba2)
        )
)
```

This is a description of a formation of five Roomba robots arranged in a cross shape. The *max-dist* relation is used to bound formation in space by keeping all robots close one to another.
Example below describes diamond shape formed by team of eight Roomba robots.

Example 4.

```
(diamond
        (set
                (max-dist 0.11 roomba1 (between roomba1 roomba0 roomba2))
                (max-dist 0.11 roomba3 (between roomba3 roomba1 roomba4))
                (max-dist 0.11 roomba5 (between roomba5 roomba4 roomba6))
                (max-dist 0.11 roomba7 (between roomba7 roomba0 roomba6))
                (between roomba1 roomba0 roomba2)
                (between roomba1 roomba0 roomba3)
                (between roomba1 roomba2 roomba7)
                (between roomba1 roomba3 roomba7)
                (between roomba3 roomba2 roomba4)
                (between roomba3 roomba2 roomba5)
                (between roomba3 roomba1 roomba5)
                (between roomba3 roomba1 roomba4)
                (between roomba5 roomba4 roomba6)
                (between roomba5 roomba4 roomba7)
                (between roomba5 roomba3 roomba7)
                (between roomba5 roomba3 roomba6)
                (between roomba7 roomba0 roomba6)
                (between roomba7 roomba0 roomba5)
                (between roomba7 roomba1 roomba5)
                (between roomba7 roomba1 roomba6)
                (not-between roomba1 roomba0 roomba4)
                (not-between roomba1 roomba2 roomba6)
                (not-between roomba1 roomba2 roomba3)
                (not-between roomba3 roomba0 roomba4)
                (not-between roomba3 roomba2 roomba6)
                (not-between roomba3 roomba1 roomba2)
                (not-between roomba5 roomba6 roomba7)
                (not-between roomba5 roomba2 roomba6)
                (not-between roomba5 roomba0 roomba4)
                (not-between roomba7 roomba5 roomba6)
                (not-between roomba7 roomba2 roomba6)
                (not-between roomba7 roomba0 roomba4)
                (not-between roomba0 roomba1 roomba5)
                (not-between roomba0 roomba3 roomba7)
                (not-between roomba2 roomba1 roomba5)
                (not-between roomba2 roomba3 roomba7)
                (not-between roomba4 roomba1 roomba5)
                (not-between roomba4 roomba3 roomba7)
                (not-between roomba6 roomba1 roomba5)
                (not-between roomba6 roomba3 roomba7)
        )
)
```

If formation is malformed our driver can try to repair it. A run-time parameter of the driver indicates which one of three methods should be used to move robots into their desired positions within the formation.

8.2.1. Three methods of formation repairing

We propose three methods for restoring a team to its prescribed formation shape. The first method is behavioral and does not use any planning. The second one is decoupled as planning is made for each robot separately, and global as all robots are taken into consideration at the same time. The third method is decoupled and global, and in addition is behavioral, as all robots move simultaneously.

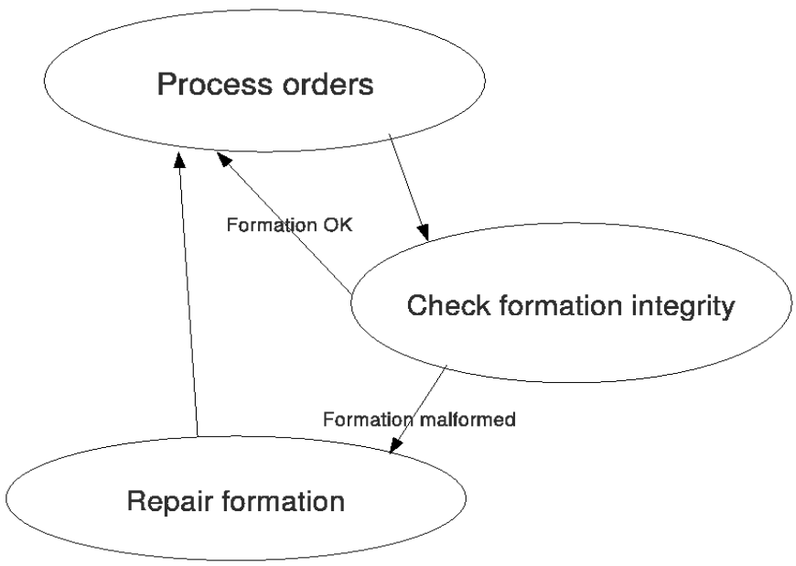

Fig. 6. States of our formation keeping driver for Player server

Method 1. **Pure behavioral**. Each robot (except a selected leader) moves to the goal position. Whenever collision is detected (on the robot bumper device), robot goes back for a while then turns left or right for a while and from this new situation, it tries again to go towards goal position. Due to the nature of this method, formation repair process is time-consuming (reactions to possible collisions take additional time) and may be even impossible. Formation is repaired relatively to one selected member of a team called a leader (therefore this selected member sticks in place while all other robot moves to their positions). If formation is not repaired after some grace time, a next member of a team is selected to be the new leader (therefore this new selected member sticks in place while all other robot moves which changes whole situation). If there are no members left to be new leaders, this method signals that the formation shape is impossible to be restored.

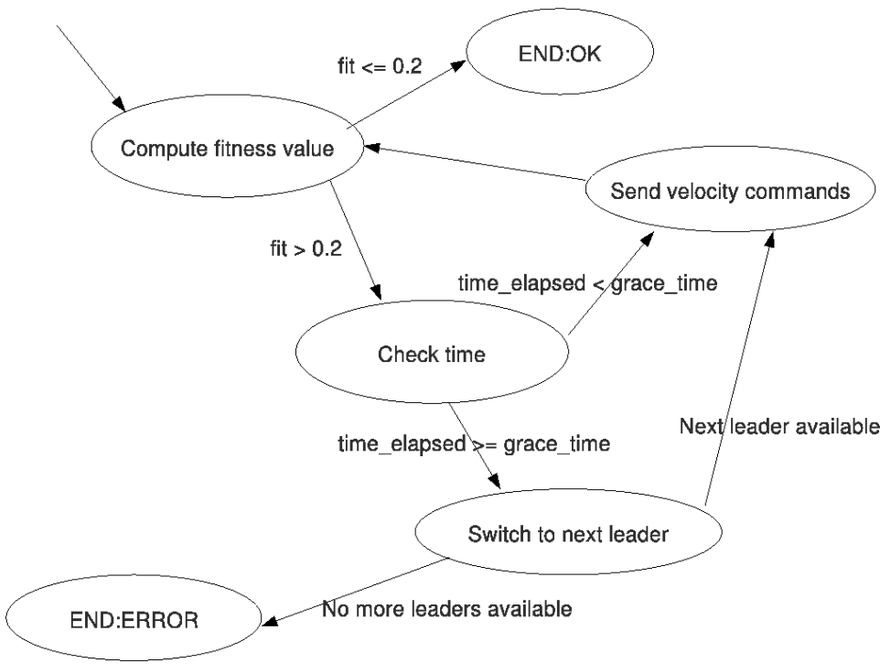

Fig. 7. Pure behavioral method of repairing the formation

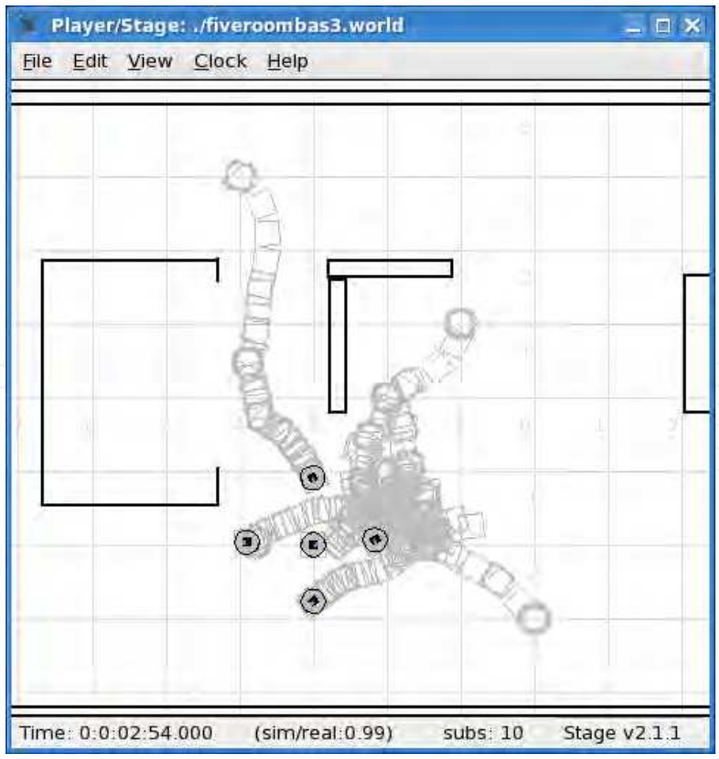

Fig. 8.Trails of robots that moved to their positions using pure behavioral method of repairing the formation

Method 2. **One robot at a time**. The procedure is repeated for each robot in a team: a path is planned by using any available planner (e.g., *wavefront* planner shipped with Player, or *mereonavigator* planner created during our experimentations (Osmialowski, 2009)); then, a robot moves to the goal position. This is the most reliable method, however it is time too consuming for bigger teams.

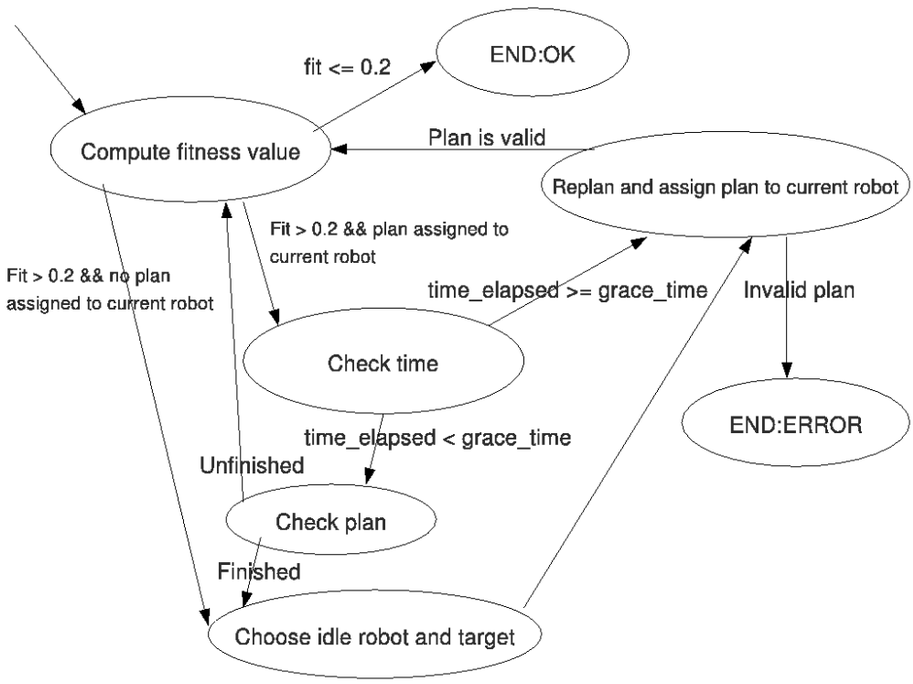

Fig. 9. *One robot at a time* method for repairing the formation

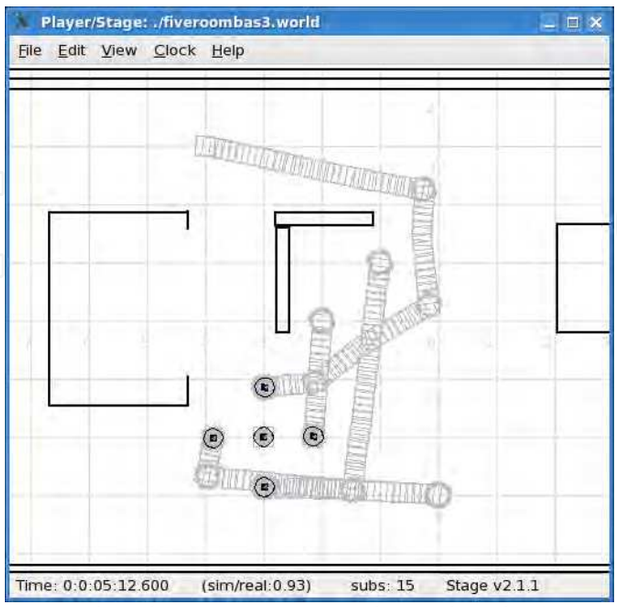

Fig. 10. Trails of robots moved to their positions on planned paths (*one robot at a time* method)

Method 3. All robots at a time. Paths are planned and executed for all robots simultaneously. Whenever collision occurs during plan execution, lower level behavior causes involved robots to go back for a while, turn left or right for a while and new paths for those robots are planned. This is the fastest method and despite the fact that it is not collision aware, it is reliable enough.

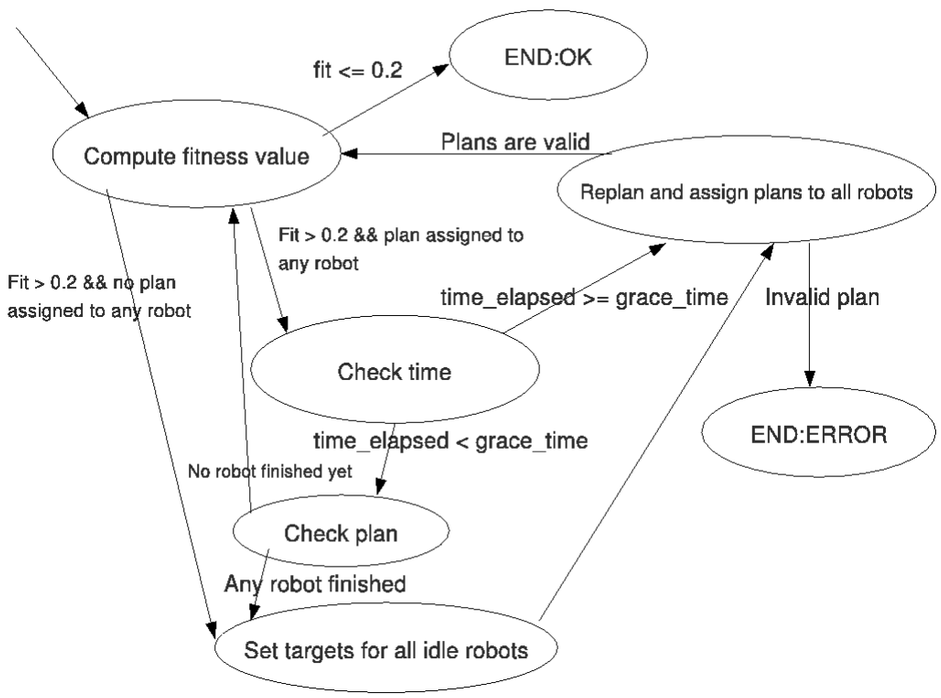

Fig. 11. All robots at a time method of repairing the formation

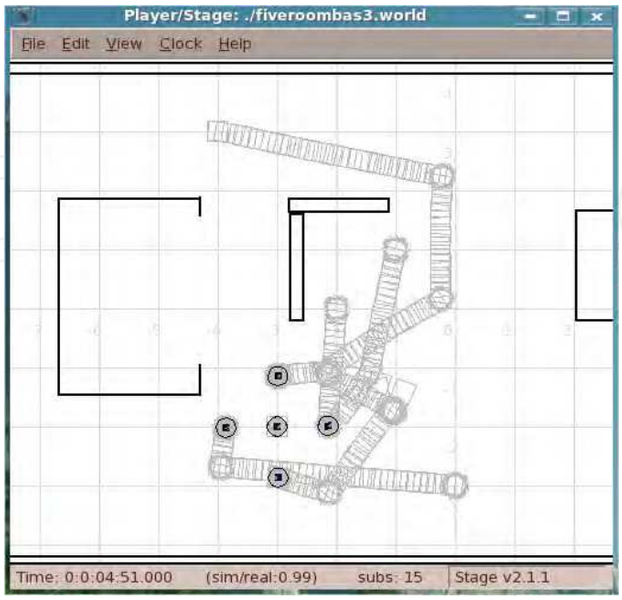

Fig. 12. Trails of robots moved to their positions on planned paths (*all robots at a time* method)

9. Navigation by obstacles with robot formations. Formation changing and repairing

The final stage of planning is in checking its soundness by navigating robots in an environment with obstacles. We show results of navigating with a team of robots in the initial formation of cross-shape in a crowded environment, see Fig. 13. In order to bypass a narrow avenue between an obstacle and the border of the environment, the formation changes to a line, and after bypassing it can use repairing to restore to the initial formation (if it is required), see Figs.14-18. The initial cross-shaped formation is shown in Fig. 13 along with obstacles in the environment.

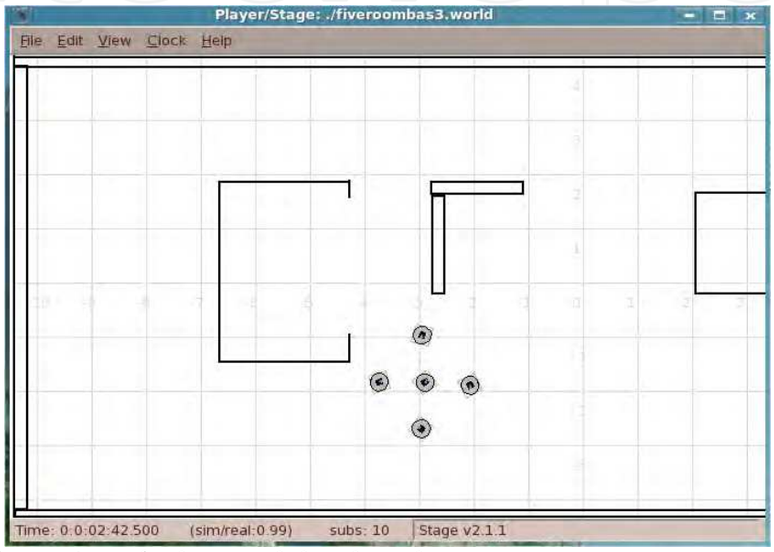

Fig. 13. Initial formation of robots and the obstacle map

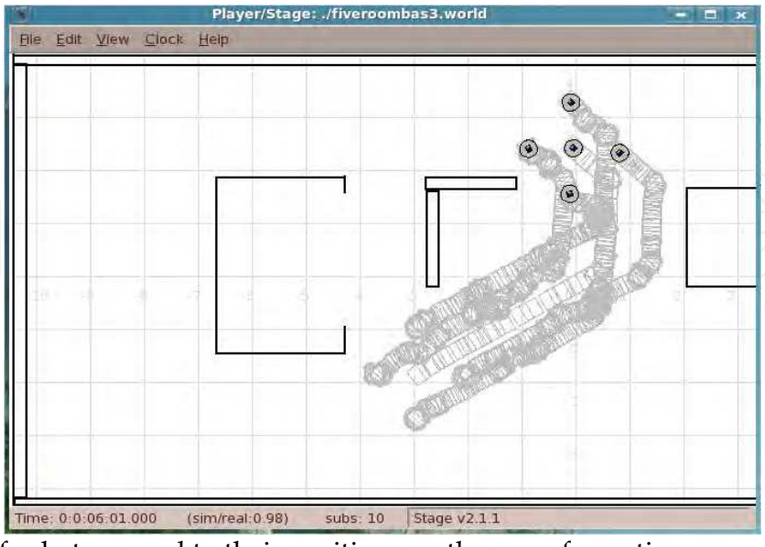

Fig. 14. Trails of robots moved to their positions on the cross formation

Reaching the target requires passing by a narrow passage between the border and the rightmost obstacle. To carry out this task, robots in the formation are bound to change the initial formation. They try the line formation, see Figs. 14-15.

However, making the line formation at the entrance to narrow passage is coupled with some difficulties: when the strategy *all robots at a time* is applied, robots at the lower part of the formation perceive robots at the upper part as obstacles and wait until the latter move into passage, which blocks whole routine as it is assumed that from the start each robot has a plan until it does reach goal or until it does collide with another robot.

To avoid such blocking of activity the behavior *wander* was added, see clouded area in Figs. 15, 16, which permitted robots to wander until they find that they are able to plan their paths into the line. It can be observed that this wandering consumes some extra time. When the strategy *one robot at a time* is applied it is important to carefully select the order in which robots are moved: the robots that have a clear pass to their target positions should go first.

Surprisingly, *pure behavioral* strategy showed good performance in managing with these difficulties, however, (as we expected) when this strategy is applied, it is time -consuming to reshape the formation. After the line was formed and robots passed through the passage, see Figs. 17-18, the line formation could be restored to the initial cross-shape, if necessary, with the help of a strategy for repairing formations of section 8.2.1. The results presented in Figs. 14-19 have been witnessing that our approach has proved its usefulness and validity: in quite complicated obstacle-ridden environments, robots are able to reach the goal. The important feature of this approach is the invariance of the notion of formation with respect to metric relations among robots: as no metric constraint bounds robots, they are able to disperse when facing an obstacle with the only requirement being to keep spatial relationships as set by the betweenness relation imposed upon them.

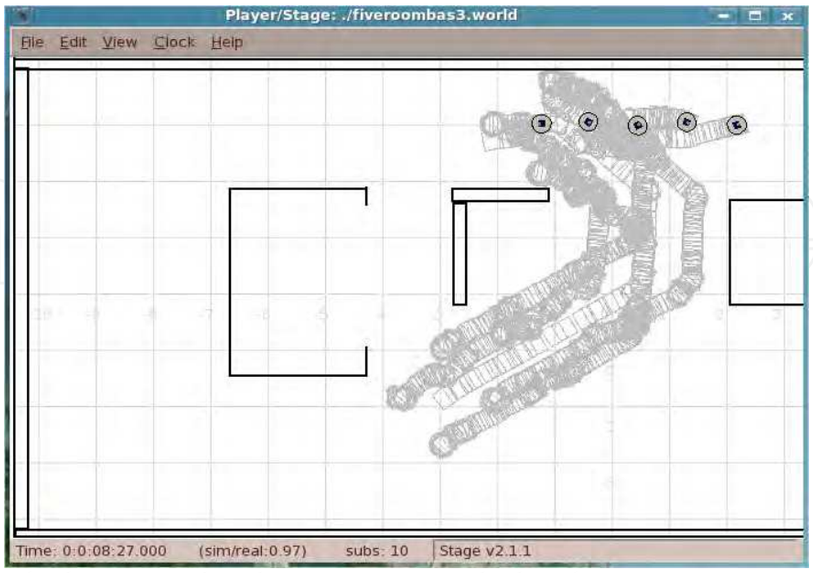

Fig. 15. Trails of robots moved to their positions on the line formation

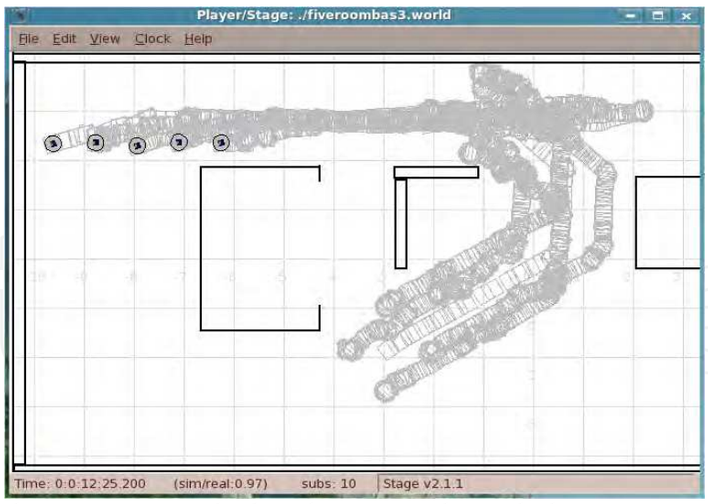

Fig. 16. Trails of robots moving in the line formation through the narrow passage

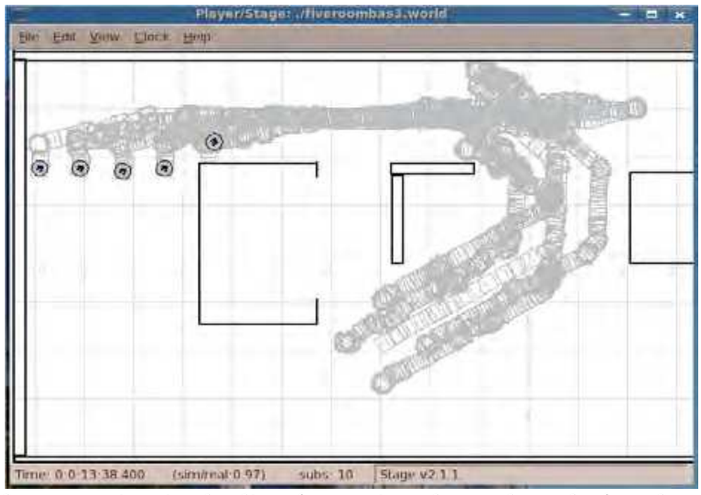

Fig. 17.Trails of robots moving in the line formation through and after the passage

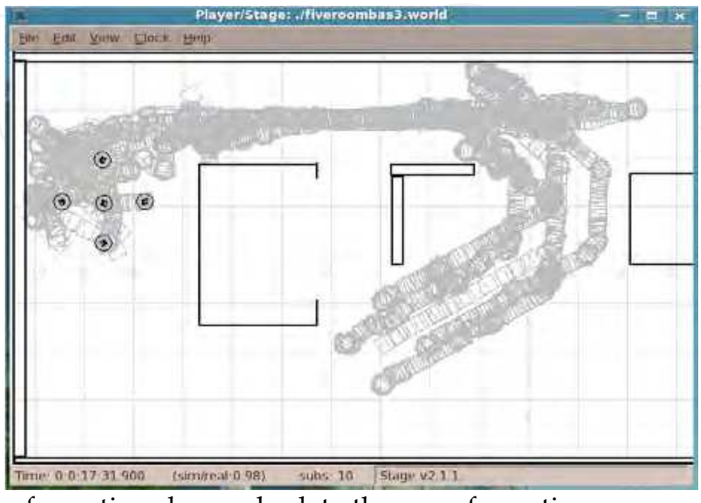

Fig. 18. Yet another formation change: back to the cross formation

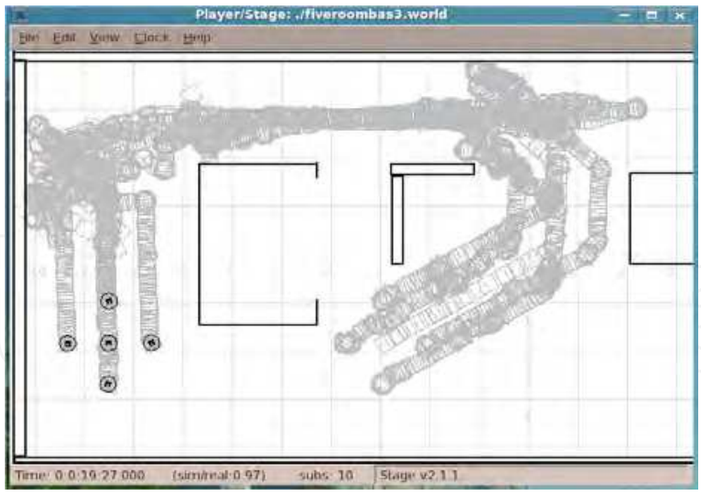

Fig. 19. Trails of robots in the cross formation in the free workspace after the passage

10. Conclusions and future research

We have proposed a precise formal definition of a formation and we have presented a Player driver for making formations according to our definition. Our definition of a formation is based on a set of rough mereological predicates which altogether define a geometry of the space. The definition of a formation is independent of a metric on the space and it is invariant under affine transformations. We have examined three methods of formation restoring, based on a reactive (behavioral) model as well as on decoupled way of planning. We have performed simulations in Player/Stage system of planning paths for formations with formation change. The results show the validity of the approach. Further research will be directed at improving the effectiveness of execution by studying divisions into sub-formations and merging sub-formations into formations as well as extending the results to dynamic environments.

11. References

Arkin, R.C. (1998). *Behavior-Based Robotics*, MIT Press, ISBN 0-262-01165-4 , Cambridge MA

Balch, T. & Arkin, R.C. (1998). Behavior-based formation control for multi-robot teams. *IEEE Transactions on Robotics and Automation*, 14(6), 926-939, ISSN 1042-296X

vanBenthem, J. (1983). *The Logic of Time*, Reidel, ISBN 9-027-71421-5, Dordrecht

Brumitt, B.; Stentz, A.; Hebert, M. & CMU UGV Group. (2001). Autonomous driving with concurrent goals and multiple vehicles: Mission planning and architecture. *Autonomous Robots*, 11, 103-115, ISSN 0929-5593

Uny Cao, Y.; Fukunaga, A.S. & Kahng, A.B. (1997). Cooperative mobile robotics: Antecedents and directions. *Autonomous Robots*, 4, 7-27, ISSN 0929-5593

Chen, Q. & Luh, J. Y. S. (1998). Coordination and control of a group of small mobile robots, *Proceedings of IEEE Intern. Conference on Robotics and Automation*, pp. 2315-2320, ISBN 0-7803-4300-X, Leuven, May 1998, IEEE Press

Choset, H.; Lynch, K.M.; Hutchinson, S.; Kantor, G.; Burgard, W.; Kavraki, L.E.& Thrun, S. (2005). *Principles of Robot Motion. Theory, Algorithms, and Implementations*, MIT Press, ISBN 0-262-03327-5, Cambridge MA

Clarke, B. L. (1981). A calculus of individuals based on connection. *Notre Dame Journal of Formal Logic*, 22(2), 204-218, ISSN 0029-4527

Das, A.; Fierro, R.; Kumar, V.; Ostrovski, J.P.; Spletzer, J. & Taylor, C.J. A vision-based formation control framework. *IEEE Transactions on Robotics and Automation*, 18(5), 813-825, ISSN 1042-296X

Gotts, N.M.; Gooday, J.A. & Cohn, A.G. (1996). A connection based approach to common—sense topological description and reasoning. *The Monist* , 79(1), 51-75

Khatib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots. *International Journal of Robotic Research* , 5, 90-98, ISSN 0278-3649

Kramer, J. Scheutz, M. (2007). Development environments for autonomous mobile robots: A survey. *Autonomous Robots*, 22, 101-132, ISSN 0929-5593

Krogh, B. (1984). A generalized potential field approach to obstacle avoidance control. *SME-I Technical paper MS84-484*, Society of Manufacturing Engineers, Dearborn MI

Kuipers, B.J. & Byun, Y.T. (1987). A qualitative approach to robot exploration and map learning, *Proceedings of the IEEE Workshop on Spatial Reasoning and Multi-Sensor Fusion*, pp. 390-404, St. Charles IL, October 1987, Morgan Kaufmann, San Mateo CA

Ladanyi, H. (1997). SQL Unleashed, Sams Publishing, ISBN 0-672-31133-X

De Laguna, T. (1922). Point, line, surface as sets of solids. *J. Philosophy,* 19, 449-461, ISSN 0022-362-X

Latombe, J. (1991). *Robot Motion Planning*, Kluwer, ISBN 0-792-39206-X, Boston

Ehrich Leonard, N. & Fiorelli, E. (2001). Virtual leaders, artificial potentials and coordinated control of groups, *Proceedings of the 40th IEEE Conference on Decision and Control*, ISBN 0-780-37061-9, pp. 2968-2973, Orlando Fla, December 2001, IEEE Press

Leonard, H. & Goodman, N. (1940). The calculus of individuals and its uses. *The Journal of Symbolic Logic,* 5, 45-55, ISSN 0022-4812

Lesniewski, S. (1916). *O Podstawach Ogolnej Teorii Mnogosci* (*On Foundations of General Theory of Sets*, in Polish), The Polish Scientific Circle in Moscow, Moscow

Lesniewski, S. (1982). On the foundations of mathematics, *Topoi* 2, 7-52, ISSN 0167-7411

Osmialowski, P. (2007). Player and Stage at PJIIT Robotics Laboratory, *Journal of Automation, Mobile Robotics and Intelligent Systems*, 2, 21-28, ISSN 1897-8649

Osmialowski, P. (2009). On path planning for mobile robots: Introducing the mereological potential field method in the framework of mereological spatial reasoning, *Journal of Automation, Mobile Robotics and Intelligent Systems*, 3(2), 24-33, ISSN 1897-8649

Osmialowski, P. & Polkowski, L. (2009). Spatial reasoning based on rough mereology: path planning problem for autonomous mobile robots, *Transactions on Rough Sets. Lecture Notes in Computer Science*, in print, ISSN 0302-9743, Springer Verlag, Berlin

Player/Stage: Available at http://playerstage.sourceforge.net

Polkowski,L. (2001). On connection synthesis via rough mereology, *Fundamenta Informaticae*, 46, 83-96, ISSN 0169-2968

Polkowski, L. (2008). A unified approach to granulation of knowledge and granular computing based on rough mereology: A survey, In: *Handbook of Granular Computing*, Pedrycz, W.; Skowron, A. & Kreinovich, V., (Eds.), 375-400, John Wiley and Sons, ISBN 987-0-470-03554-2, Chichester UK

Polkowski, L. & Osmialowski, P. (2008) Spatial reasoning with applications to mobile robotics, In: *Mobile Robots Motion Planning. New Challenges,* Xing-Jian Jing, (Ed.), 43-55, I-Tech Education and Publishing KG, , ISBN 978-953-7619-01-5, Vienna

Polkowski, L. & Osmialowski, P. (2008). A framework for multiagent mobile robotics: Spatial reasoning based on rough mereology in Player/stage system, *Lecture Notes in Artificial Intelligence,* 5306, 142-149, Springer Verlag, ISSN 0302-9743, Berlin

P. Ramsey, PostGIS Manual, in: postgis.pdf file downloaded from Refractions Research home page.

J. Shao, G. ; Xie, J. Yu & Wang, L. (2005). Leader-following formation control of multiple mobile robots, In: *Proceedings of the 2005 IEEE Intern. Symposium on Intelligent Control*, pp. 808-813, ISBN 0-780-38936-0, Limassol, June 2005, IEEE Press

sfsexp: Available at http://sexpr.sourceforge.net

Sugihara, K. & Suzuki, I. (1990). Distributed motion coordination of multiple mobile robots, In: *Proceedings 5th IEEE Intern. Symposium on Intelligent Control*, pp. 138-143, ISBN 9-991-32943-9, Philadelphia PA, Sept. 1990, IEEE Press

Švestka, P. & Overmars, M.H. (1998). Coordinated path planning for multiple robots, *Robotics and Autonomous Systems*, 23, 125-152, ISSN 0921-8890

Tarski, A. (1929) Les fondements de la géométrie des corps, In: *Supplement to Annales de la Sociéte Polonaise de Mathématique*, 29-33, Krakow, Poland

Tarski, A. (1959). What is elementary geometry?, In: *The Axiomatic Method with Special Reference to Geometry and Physic*s, Henkin, L.; Suppes, P. & Tarski, A., (Eds)., 16-29, North-Holland, Amsterdam

Tribelhorn, B. & Dodds, Z. (2007). Evaluating the Roomba: A low-cost, ubiquitous platform for robotics research and education, In: *2007 IEEE International Conference on Robotics and Automation, ICRA 2007*, pp. 1393-1399, ISBN 1-4244-0601-3, April 2007, Roma, Italy, IEEE Press

Urdiales, C.; Perez, E.J.; Vasquez-Salceda, J.; S*a*nchez-Marrµe, M. & Sandoval, F. (2006). A purely reactive navigation scheme for dynamic environments using Case-Based Reasoning, *Autonomous Robots* , 21, 65-78, ISSN 0929-5593

Whitehead, A. N. (1979). *Process and Reality. An Essay in Cosmology* 2nd. ed., The Free Press, ISBN 0-029-34570-7, New York NY

Mobile Robots Navigation

Edited by Alejandra Barrera

ISBN 978-953-307-076-6
Hard cover, 666 pages
Publisher InTech
Published online 01, March, 2010
Published in print edition March, 2010

Mobile robots navigation includes different interrelated activities: (i) perception, as obtaining and interpreting sensory information; (ii) exploration, as the strategy that guides the robot to select the next direction to go; (iii) mapping, involving the construction of a spatial representation by using the sensory information perceived; (iv) localization, as the strategy to estimate the robot position within the spatial map; (v) path planning, as the strategy to find a path towards a goal location being optimal or not; and (vi) path execution, where motor actions are determined and adapted to environmental changes. The book addresses those activities by integrating results from the research work of several authors all over the world. Research cases are documented in 32 chapters organized within 7 categories next described.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen