2007 IEEE International Conference on
Robotics and Automation
Roma, Italy, 10-14 April 2007



# RFID-Based Exploration for Large Robot Teams

V. A. Ziparo*, A. Kleiner*, B. Nebel* and D. Nardi**

*Abstract*— **To coordinate a team of robots for exploration is a challenging problem, particularly in large areas as for example the devastated area after a disaster. This problem can generally be decomposed into task assignment and multi-robot path planning. In this paper, we address both problems jointly. This is possible because we reduce significantly the size of the search space by utilizing RFID tags as coordination points.**

**The exploration approach consists of two parts: a stand-alone distributed local search and a global monitoring process which can be used to restart the local search in more convenient locations. Our results show that the local exploration works for large robot teams, particularly if there are limited computational resources. Experiments with the global approach showed that the number of conflicts can be reduced, and that the global coordination mechanism increases significantly the explored area.**

## I. INTRODUCTION

To coordinate a team of robots for exploration is a challenging problem, particularly in large areas as for example the devastated area after a disaster. This problem can generally be decomposed into task assignment and multi-agent path planning. Whereas in the context of exploration the task assignment problem has been intensively studied, there has been only little attention on avoiding conflicts in paths for large robot teams. This is mainly due to the fact that the joint state space of the planning problem grows enormously in the number of robots. However, particularly in destructed environments, where robots have to overcome narrow passages and obstacles, path coordination is essential in order to avoid collisions.

The basic approach proposed in this paper is to reduce significantly the size of the search space by utilizing RFID tags as coordination points. Robots deploy autonomously tags in the environment, in order to build a network of reachable locations. Hence, global path planning can be carried out on a graph structure, which is computationally cheaper than planning on global grid maps, as it is usually the case.

Our systems solves the problem of task assignment and path planning simultaneously. This is carried out by a two-layered approach. Firstly, a *local part*, where robots are coordinated via RFID chips and perform a local search. The local approach has the properties that the computational costs do not grow with the number of robots and that it does not need direct communication. Secondly, based on the local part, a *global part*, which is in charge of monitoring the local exploration, possibly restarting it in more convenient locations significantly improving its performance. The locations where to move the robots, and the multi-robot plan to reach them, are found solving a task assignment and planning problem.

In previous work we conducted experiments on real robots in order to evaluate whether it is feasible to autonomously deploy and detect RFID tags in a structured environment [11]. The experiments were conducted in two phases. In the first, the robot autonomously explored an unknown cellar environment while deploying successfully 50 RFID tags with its deploy device. In the second, the robot's mission was again to explore the same environment, however, to identify tags previously deployed in the environment (see [12] for a video). Furthermore, the number of retrieved tags was sufficient to reasonably correct the robot's noisy odometry trajectory.

Experiments in this work have been carried out in the USARSim [2] simulation environment, which serves as basis for the *Virtual Robots* competition at RoboCup. Our results show that the RFID tag-based exploration works for large robot teams, particularly if there are limited computational resources. Furthermore, we evaluated the global approach on RFID graphs of different complexity and size. Finally, we evaluated the full system in qualitative experiments on USARSim. Our results show that the number of conflicts can be reduced by sequence optimization, and that this global coordination mechanism combined with the local approach, increases significantly the explored area.

Methods for local exploration have already been successfully applied in the past [3], [16]. It has basically been shown that multi-robot terrain coverage is feasible without robot localization and an exchange of maps. Burgard and colleagues [5] contributed a method for greedy task assignment, based on grid mapping and frontier cell exploration [17]. Their method does not consider conflicts between single robot plans, and requires robots to start their mission close to each other with knowledge about their initial displacement. The work by Bennewitz and colleagues [4] focuses on the optimization of plans taken by multiple robots at the same time. They select priority schemes by a hill-climbing method that decides in which

*Department of Computer Science, University of Freiburg, Georges-Köhler-Allee 52, D-79110 Freiburg, Germany, {ziparo,kleiner,nebel}@informatik.uni-freiburg.de

**Dipartimento di Informatica e Sistemistica, University of Rome "Sapienza", Via Salaria 113, 00198 Rome, Italy, nardi@dis.uniroma1.it

order robots plan to their targets [8]. Plans are generated in the configuration time space by applying A* search on grid maps. The coordinated movement of a set of vehicles has also been addressed in other domains, such as in the context of operational traffic control [9], and the cleaning task problem [10].

The remainder of this paper is structured as follows. In Section II we describe the system platform. In Sections III and IV we respectively describe our local and global approach for coordinated exploration. Finally, we provide results from experiments in Section V and conclude in Section VI.

## II. TEST PLATFORM

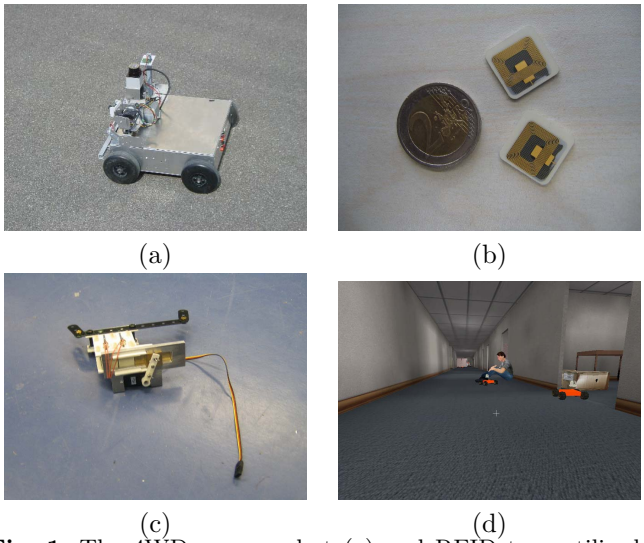

**Fig. 1.** The 4WD rescue robot (a) and RFID tags utilized with this robot (b) with a hand crafted deploy device (c). A model of this robot simulated in the USARSim simulator within an office-like environment (d).

The test platform utilized for experiments presented in this paper is based on a four wheel drive (4WD) differentially steered robot, as depicted in Figure 1(a). The robot is equipped with a *Hokuyo* URG-X003 Laser Range Finder (LRF), and an Inertial Measurement Unit (IMU) from *XSense* providing measurements of the robot's orientation by the three Euler angles *yaw*, *roll*, and *pitch*. We utilized Ario RFID chips from *Tagsys* (see Figure 1(b)) with a size of $1.4 \times 1.4cm$, $2048Bit$ RAM, and a response frequency of 13.56MHz. They implement an anti-collision protocol, which allows the simultaneous detection of multiple RFIDs within range. For the reading and writing of the tags we employed a Medio S002 reader, likewise from *Tagsys*, which allows to detect the tags within a range of approximately $30cm$ while consuming less than $200mA$. The antenna of the reader is mounted in parallel to the ground. This allows to detect any RFID tag lying beneath the robot. The active distribution of the tags is carried out by a self-constructed actuator, realized by a magazine, maximally holding 100 tags, and a metal slider that can be moved by a conventional servo. Each time the mechanism is triggered, the slider moves back and forth while dropping a single tag from the magazine.

A realistic model of the robot, including the RFID tag release device, is simulated with the USARSim simulator developed at the University of Pittsburgh [6], [2]. USARSim allows a real-robot simulation of raw sensor data, which can directly be accessed via a TCP/IP interface. The sensors of the robot model are simulated with the same parameters as the real sensors, expect the real RFID reading and writing range. Without loss of generality, we set this range to two meters, since this parameter mainly depends on the communication frequency and size of the transmitter's antenna, which both can be replaced.

## III. COORDINATED LOCAL EXPLORATION

In this section we present a coordination mechanism which allows robots to explore an environment with low computational overhead and communication constraints. In particular, the computational costs do not increase with the number of robots. The key idea is that the robots plan their path and explore the area based on a local view of the environment, where consistency is maintained through the use of indirect communication, i.e. RFIDs.

### A. Navigation

To efficiently and reactively navigate, each robot continuously path plans based on its local information of the environment, which is maintained within an occupancy grid. This representation of the environment, for allowing fast computation, is limited in size. In particular, in our implementation, we restricted it to a four meter side square with forty mm resolution. The occupancy grid is shifted based on the odometry and continuously updated based on new scans. This avoids the accumulation of the odometry errors when moving, while having some memory of the past. We periodically select a target, as shown in the next subsection, and produce for it plans at high frequency. The continuous re-planning allows to reactively avoid newly perceived obstacles or unforeseen situations caused by errors in path following.

The path planning algorithm is based on A* [15] with the Euclidean distance heuristic. We expand all the neighbors of a cell which are not obstructed (i.e. have an occupancy value lower than a given threshold). The cost function $c$ takes into account the length of the path and the vicinity of the obstacles to the path in the following way:

$$c(s_{i+1}) = c(s_i) + d(s_{i+1}, s_i) * (1 + \alpha * occ(s_{i+1})) \quad (1)$$

where $occ(s)$ is the current value of the occupancy grid in cell $s$, $d(.)$ is the distance, and $\alpha$ is a factor for varying the cost for passing nearby obstacles. Before planning, the grid is convoluted with a Gaussian kernel, which allows to keep robots as far as possible from obstacles.

While navigating in the environment, robots maintain a local RFIDs set (LRS), which contains all the perceived RFIDs which are in the range of the occupancy grid. On the basis of this information, new RFIDs are released in the environment by the robots in order to maintain a predefined density of the tags (in our implementation we take care of having the RFIDs at one meter distance from each other). Note that nowadays most of the RFID tags available on the market do implement an anti-collision protocol, and hence the detection of multiple RFIDs is possible at the same time. We utilize the local knowledge that robots have on RFID tags for avoiding collisions between them. Each robot tracks its own pose by integrating measurements from the wheel odometry and IMU sensor with a Kalman filter. As commonly known, the accuracy of this estimate decreases due to the accumulation of positioning errors, which can, for example, be prevented by performing data association with visual features. However, since our goal is to save computation time, we do not globally improve the pose estimate during runtime, instead we synchronize the local displacement between robots via RFID tags. If two robots have visited the same RFID tag in the past, the estimates of their mutual displacement $d_{R_1R_2} \approx l_{R_1} - l_{R_1}$ can be synchronized by utilizing their local pose estimates at this RFID tag: Let $l_{R_1}(t_1)$ and $l_{R_2}(t_2)$ denote the individual pose estimates of robot $R_1$ and $R_2$ while visiting the same RFID tag at time $t_1$ and time $t_2$, respectively. Then, the new displacement between both robots can be calculated by $d_{R_1R_2} = l_{R_1}(t_1) - l_{R_1}(t_2)$. Furthermore, each robot can estimate poses within the reference frame of other robots by utilizing the latest displacement and the individual pose estimate of the other robot at time $t$. For example, $R_2$'s pose estimate of $R_1$ is given by: $\hat{l}_{R_1}(t) = l_{R_1}(t) - d_{R_1R_2}$. Note that this procedure assumes the existence of a synchronized clock and requires the robots to keep their trajectory in memory.

The knowledge on the poses of other robots enables to avoid collisions among teammates. This is carried out by labeling occupancy grid cells within a given range from the teammate as penalized, which will be taken into account at the planning level by adding an extra cost for going through such locations. If a robot detects that a teammate with a higher priority (which is predefined) is closer than a security distance it stops until this has moved out of the way.

### B. Local Exploration

The fundamental problem in the local exploration task is how to select targets for the path planner in order to minimize overlapping of explored areas. This involves: i) choosing a set of target locations $F = \{f_j\}$, ii) computing an utility value $u(f_j)$ for each target location $f_j \in F$ and iii) selecting the best target, based on the utility value, for which the path planner can find a trajectory.

We first identify a set of targets $F$ by extracting frontiers $F$ [17] from the occupancy grid. We then order the set based on the following utility calculation:

$$u(f_j) = -\gamma_1 * angle(f_j) - \gamma_2 * visited(f_j) \qquad (2)$$

where $angle(f_j)$ is a value which grows quadratically with the angle of the target with respect to the current heading of the robot. The angle factor can be thought as an inertial term, which prevents the robot from changing too often direction (which would result in an inefficient behavior). If the robot would have full memory of his perceptions (i.e. a global occupancy grid), the angle factor would be enough to allow a single robot to explore successfully. Due to the limitation of the occupancy grid, the robot will forget the areas previously explored and thus will possibly go through already explored ones.

In order to maintain a memory of the previously explored areas the robots store in the nearest RFID at writing distance poses $p$ from their trajectory (discretized at a lower resolution respect to the occupancy grid). The influence radius, e.g. the maximal distance in which poses are added, depends mainly on the memory capacity of the RFID tag. In our implementation, poses where added within a radius of 4 meters. Moreover, a value $count(p)$ [16] is associated with each pose $p$ in the memory of the RFID and is incremented by the robots every time the pose is added. These poses $p$ are then used to compute $visited(f_j)$ as $\sum_{r \in LRS} \sum_{p \in P_r} (1/d(f_j, p)) * count(p)$, where $P_r$ is the set of poses associated with the RFID $r$.

Finally, $\gamma_1$ and $\gamma_2$ are two parameters which control the trade-off between direction persistence and exploration. It is worth noticing that robots writing and reading from RFIDs, not only maintain memory of their own past but also of the other robots implementing a form of indirect communication. Thus, both multi-robot navigation and exploration, do not require direct communication. This feature is very useful in all those scenarios (e.g. disaster scenarios) where wireless communication may be limited or unavailable.

The most important feature of the approach, as presented up to now, is that the computation costs do not increase with the number of robots. Thus, in principle, there is no limit, other than the physical one, to the number of robots composing the team.

## IV. GLOBAL EXPLORATION MONITORING

Due to the lack of lookahead of the local exploration, robots may last too long in local minima, resulting in useless coverage of already explored areas. In order to avoid such a phenomenon, a novel monitoring approach has been developed, which periodically restarts the local exploration in more convenient locations. This method requires direct communication and a computational overhead, which grows with the number of agents. However, it greatly improves the exploration ability of the robots and it is robust to failures. In fact, if the communication

links fail or the monitoring process itself fails, the robots will continue the local exploration previously described.

### A. Problem Modeling

Basically, the problem is to find a target RFID location for each robot and a multi-robot path for them. We assume that an RFID graph $G = (V, E)$, where $V$ is the set of RFID positions and $E$ passable links between them, is available. Each node consists of a unique identifier for the RFID and its estimated position. Moreover, a set of frontier nodes $U \subset V$ and a set of current robot RFID positions $SL \subset V$ is defined. In general, $|U| > |R|$, where $R$ is the set of available robots. A robot path (i.e plan) is defined as a set of couples composed by a node $v \in V$ and a time-step $t$:

**Definition IV.1** *A* single-robot plan *is a set*

$$P_i = \{<v, t> \mid v \in V \wedge t \in T\}$$

*where* $T = \{0, \ldots, |P_i| - 1\}$. $P_i$ *must satisfy the following properties:*
*a)* $\forall v_i, v_j, k \ <v_i, k> \in P_i \wedge <v_j, k+1> \in P_i \Rightarrow (v_i, v_j) \in E$,
*b)* $<v, 0> \in P_i \Rightarrow v = sl_i \in SL$
*c)* $<v, |P_i| - 1> \in P_i \Rightarrow v \in U$

where property a) states that each edge of the plan must correspond to an edge of the graph $G$. Properties b) and c) enforce that the first and the last node of a plan must be the location of a robot and a goal node respectively. For example, the single-robot plan going from RFID R1 to RFID G1, depicted in Figure 2 is represented as $P_1 = (< R1, 0 >, < N1, 1 >, < N2, 2 >, < G1, 3 >)$. The previous definition implies that passing any two

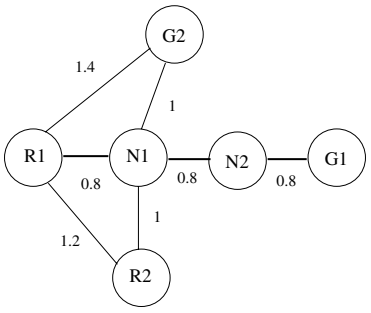


Fig. 2. A simple graph showing a plan from $R1$ to $G1$ (bold edges).

nodes, which are connected by an edge in the graph $G$, takes approximately the same amount of time. Recall that nodes represent RFIDs which are deployed approximately at the same distance one from the other, and edges represent shortest connection between them. Thus, the difference of time required for traveling between any two connected RFIDs is negligible small, if robots drive at the same speed.

**Definition IV.2** *A* multi-robot plan $P$ *is a n-tuple of single-robot plans* $(P_1, \ldots, P_n)$ *such that:*
*a) the plan with index i belongs to robot i,*
*b)* $\forall i, j \in R \ <v', |P_i| - 1> \in P_i \wedge <v'', |P_j| - 1> \in P_j \Rightarrow v' \neq v''$
*c)* $\forall i, j \in R \ <v', 0> \in P_i \wedge <v'', 0> \in P_j \Rightarrow v' \neq v''$

Thus, a multi-robot plan is a collection of single-robot plans for each robot such that they all have different goals and different starting positions. A distinguishing feature of multi-robot plans with respect to single-agent ones is interaction. In fact, single-robot plans can interfere with each other leading to inefficiencies or even failures:

**Definition IV.3** *Two single-robot plans* $P_i$ *and* $P_j$ *of a multi-robot plan* $P$ *are said to be in* conflict *if* $P_i \cap P_j \neq \emptyset$. *The set of states* $C_{P_i} = \bigcup_{i \neq j} P_i \cap P_j$ *are the conflicting states for* $P_i$.

Moreover, *deadlocks* can occur in the system. Due to space limitations we omit the formal definition. In our setting, a deadlock can arise if there is a circular wait or if a robot is willing to move to an already achieved goal of another robot. Consequently, the cost measure $c(.)$ for a multi-robot plan $P$ is defined as follows:

$$c(P) = \begin{cases} \infty & \text{if deadlock} \\ \max\limits_{i \in R} cost(P, i) & \text{else} \end{cases} \quad (3)$$

where $cost(P, i)$ is the cost of executing $i$'s part of the multi-robot plan $P$. We assume that the agents execute the plans in parallel, thus the score of the multi-robot plan is the maximum among the single-robot ones. Let $P_j(t)$ be a function that returns the RFID node of a plan $P_j$ at a time index $t$, and $d(.)$ the Euclidean distance between two RFIDs. Then, $cost(P, i)$ can be computed from the sum of the Euclidean distances between the RFIDs of the plan plus the conflicts cost:

$$cost(P, i) = \sum_{t=0}^{|P_i|-2} d(P_i(t), P_i(t+1)) + confl(P, i) \quad (4)$$

where

$$confl(P, i) = \sum_{j \neq i} \sum_{<v, t> \in P_i \cap P_j} wait(P_j, t), \quad (5)$$

and

$$wait(P_j, t) = d(P_j(t-1), P_j(t)) + d(P_j(t), P_j(t+1)), \quad (6)$$

whereas the wait cost $wait(P_j, t)$ reflects the time necessary for robot $j$ to move away from the conflict node. By Equation 5 costs for waiting are added if at least two robots share the same RFID node at the same time. This is a worst case assumption, since conflicts in the final multi-robot plan are solved by the local coordination mechanism which forces robots only to wait if there are other robots with higher priority. We abstract this feature from our model since the priority ordering is periodically randomized in order to solve existing deadlocks, making it impossible to predict whether a robot will have to wait or not. Finally, the Task Assignment and Path Planning problem can be formulated as an optimization problem of finding a plan $P^*$ that minimizes the cost function $c(P)$.

*B. Global Task Assignment and Path Planning*

We experimented with three different techniques in order to solve the Task Assignment and Path Planning problem. The first two approaches are inspired by Burgard et al. [5]. The third approach, can be seen as an extension of Bennewitz et al.[4]. All of the previously cited approaches rely on a grid based representation while our approach is graph-based. The experimental results show that the third approach outperforms the first and the second, and is actually the one we adopted in the implementation of the full system. For this reason, we just give a brief overview of the first approaches and detail more carefully the third.

All the approaches have a common pre-calculation. We compute the Dijkstra graph [7] for each node in $U$. This is a fast computation (i.e. $O((|E|+|V|log(|V|))|U|)$ ) which speeds up the plan generation processes presented in the following.

*1) Greedy Approach:* Given the information produced by the Dijkstra algorithm and an empty multi-robot plan, we identify the robot $r_{best} \in R$ which has the shortest path to reach a goal $g_{best} \in U$. The path computed by the Dijkstra algorithm from $r_{best}$ to $g_{best}$, with its time values, is added to the multi-robot plan. We then update the sets $R = R - \{r_{best}\}$ and $U = U - \{g_{best}\}$. The process is iterated until $R = \emptyset$ (see [5] for more details).

*2) Assignment Approach:* A common approach in multi-robot systems is task assignment. Here we utilize a genetic algorithm permuting over possible goal assignments to robots and use the plans computed by the Dijkstra algorithm. We then use the previously defined cost function as the fitness function (see [4] for more details).

*3) Sequential Approach:* The last approach we present is based on sequential planning. We use, in a similar way to the assignment approach, a genetic algorithm to permute possible orderings of agents $O = o_1, \ldots, o_n$. We then plan for the ordering and use the previously defined cost function as the fitness function.

The sequential planning is based on A* [15] and is done individually, following the given ordering, for each agent in order to achieve the most convenient of the available goals $U$. Every time an agent $o_i$ plans, the selected goal is removed form $U$ and the computed plan added to the set of known plans $P$. The planning tries to avoid conflicts with the set of known plans $P$ by searching through time/space, whereas the state space $S$ is defined as $S = V \times T$. This huge state space can be greatly simplified, since for our purposes we are only interested in the time of the conflicting states $C_{P_j}$ (Definition IV.3). From the planning point of view the information relative to the time of non-conflicting states is irrelevant and thus all these states can be grouped by time using the special symbol *none*. The resulting set of non-conflicting states $NC$ is defined as $NC = \{<v, none> \mid v \in V\}$ and has the cardinality of $V$ (i.e. $|NC| = |V|$). Thus, the reduced search space is $ST = C_{P_j} \cup NC$.

During the search, the nodes are expanded in the following way: we look for the neighboring nodes of the current one given the set $E$ of edges in $G$. For each of them we check if there is a conflict. If this is the case, we return the corresponding node from $C_{P_j}$, otherwise the one from $NC$.

In order to implement A* we have to provide a cost function $g$ and a heuristic function $h$ defined over $ST$. We define the cost function $g(s)$ for agent $i$ as the single-robot plan cost function $cost(P, i)$. The multi-robot plan $P$ will consist of the plans already computed augmented with the path found up to $s$. Obviously the agent $o_j$ will be able to detect conflicts at planning time only for those agents $o_i$ with $i < j$ for which a plan has already been produced. Finally, the heuristic $h$ (i.e. Dijkstra heuristic) is defined as follows:

$$h(< s, t >) = \min_{g \in G} d_{dij}(s, g) \tag{7}$$

where $d_{dij}(s, g)$ is the distance from $s$ to $g$ precomputed by the Dijkstra algorithm.

**Theorem IV.1** *The Dijkstra heuristic is admissible.*

*Proof:* By contradiction. Let us assume that the theorem is false and, thus, that $\exists s \in S \mid \min_{g \in G} d_{dij}(s, g) > cost(P, i)$. $P$ is the multi-robot plan composed by plans already computed plus a path $P_i$ from $s$ to a goal $g_{better}$. Assuming that $s_c$ is the state which verifies the property and that $g_{min}$ is the closet goal to $s_c$, we can rewrite the previous inequality as $d_{dij}(s_c, g_{min}) > cost(P, i)$. Applying the definition of $cost(P, i)$, we can rewrite the inequality as $d_{dij}(s_c, g_{min}) > d_{plan}(s_c, g_{better}) + confl(P, i)$; where $d_{plan}(s_c, g_{better}) = \sum_{t=0}^{|P_i|-2} d(P_i(t), P_i(t+1))$. $confl(P, i)$ is the sum of values which are distances calculated in the Euclidean space which are alway values greater or equal to zero. This implies $d_{dij}(s_c, g_{min}) > d_{plan}(s_c, g_{better})$. Since $d_{dij}(s_c, g_{min}) < d_{dij}(s_c, g_{better})$ we can rewrite the inequality as $d_{dij}(s_c, g_{better}) > d_{plan}(s_c, g_{better})$. This means that there exists a path on the graph from $s_c$ to $g_{better}$ shorter than the one the Dijkstra algorithm found, but this is impossible [7]. ■

It is important to notice that A* will find the optimal solution, since the heuristic is admissible, for a single robot plan, given a subset of already computed paths and ignoring the others (for which no plan was found yet).

For example, let us consider the simple weighted graph depicted in Figure 2. $R1$ and $R2$ are respectively the location of two robots $r1$ and $r2$. $G1$ and $G2$ are the goals. In this example, the sequence $<r1, r2>$ has been selected and $r1$ has already produced the following plan: $(<R1, 0>, <N1, 1>, <N2, 2>, <G1, 3>)$. Now $r2$ has to plan. The only remaining goal is $G2$, since $G1$ has already been selected by $r1$. At first, according to the topology

and the already defined plan for $r1$, from $<R2, none>$ the nodes $<R1, none>$ and $<N1, 1>$ can be reached. In fact, if we simulate the plan of $r1$, moving to $R1$ will incur in no conflict and would have just the cost of travelling the distance; thus $g(< R1, none >) = 1.2$. In the other case, moving to $R2$ at time 1, will conflict with $r1$ who is moving there at the same time. In this case, our model will tell us that we have to wait for $r1$ to first reach the $N1$ (with a cost of 0.8) and then leave it (with a cost of 0.8). Then, we would be able to reach $N1$ with a cost of 1. Thus the total cost of reaching $N1$ at time 1 will be $g(<N1, 1>) = 2.6$ (i.e. $0.8 + 0.8 + 1$).

Moreover, the heuristic values for these states (obtained by pre-computing the Dijkstra algorithm) are: $h(< R1, none >) = 1.4$ and $h(< N1, 1 >) = 1$. Thus, according to the well known formula $f(n) = g(n) + h(n)$, $< R1, none >$ will be selected. Similarly, nodes $< N1, none >$ and $< G2, none >$ will be expanded next and the planning process will continue until the plan $(< R2, 0 >, < R1, 1 >, < G2, 2 >)$ is found. Notice that, $< N1, none >$ is different from $< N1, 1 >$ since $r2$ will already have moved away from $N1$ at time-step 2.

### C. Monitoring Agent

The monitoring agent $MA$ constructs online the map represented as the graph $G$ and identifies the frontier RFIDs. Moreover, $MA$ will monitor the local exploration and possibly identify, with one of the previously described techniques, a multi-robot plan in order to move the robots to a location where the local exploration has better performance expectations. Due to space limitations we will give just a brief overview on this topic.

At execution time the robots send to $MA$ their RFID locations (i.e. the nearest RFID they can perceive). Every time a robot changes its RFID position from $r_i$ to $r_{i+i}$, $MA$ updates the set $SL$ of current robot locations and updates the graph as follows: $E = E \cup \{(r_i, r_{i+1})\}$ and $V = V \cup \{r_i\} \cup \{r_{i+1}\}$.

The monitoring process collects continuously information regarding the unexplored area in the vicinity of the RFIDs based on the local occupancy grid to identify the frontier RFIDs $U$. Roughly, the robot knows how many RFIDs, given the defined deployment density, should be placed per square meter and which the number perceived. Thus, can compute an estimate on how much the area is explored in the proximity of his RFID position.

$MA$ periodically evaluates the position of the robots on the graph and their distance from the frontier nodes $U$. If this value exceeds a given threshold, it stops the robots and computes a new multi-robot plan. Once a valid plan has been produced, $MA$ starts to drive the robots by assigning to each of them the next RFID prescribed by their plans. The robots path-plan from one RFID to the other using the A* path-planner on the occupancy grid and the teammate avoidance previously mentioned. If the occupancy grid path planner fails to find a plan because he cannot perceive the RFID (e.g. it was destructed) or the way is obstructed, the robot sends a failure message to the agent. The agent will consequently remove the node and its edges from the graph $G$ and re-plan. When the target RFID is reached, a task accomplished message is sent to the agent, which will assign another task or send a global plan termination message. In the latter case, the robots will start again the local exploration.

During the multi-robot plan execution, the planner monitors for unforeseen situations. For example, if a robot does not send an accomplished task message or an RFID position for a long time, it is considered lost and removed from the robot list. Moreover, plans can incur deadlocks and, although we check for them at planning time, there is no guarantee of a deadlock-free execution because we cannot predict the exact order in which the tasks will be accomplished. If a deadlock occurs at a given time, $MA$ re-plans. Finally, any time a planning phase fails, the local exploration is reactivated.

## V. EXPERIMENTS

### A. Evaluation of the local approach

The local approach has been tested in various simulated environments generated by the National Institute of Standards and Technology (NIST) on the USARSim platform. They provide both indoor and outdoor scenarios of the size bigger than $1000m^2$, reconstructing the situation after a real disaster. Since USARSim allows the simulated deployment of heterogeneous robot types within a wide range of different scenarios, it offers an ideal performance metric for comparing multi-robot systems.

On these maps, we competed against other teams, during the RoboCup'06 [1] *virtual robots competition*, where our team won the first prize [12]. In this competition, virtual teams of autonomous or tele-operated robots have to find victims within 20 minutes while exploring an unknown environment. The current version of USARSim is capable of simulating up to 12 robots at the same time.

Most of the teams applied frontier cell-based exploration on global occupancy grids. In particular: selfish exploration and map merging [13] (IUB), map merging and local POMDP planning [14] (UVA), operator-based frontier selection and task assignment (SPQR), and teleoperation (STEEL) and (GROK).

Table I gives an overview on the number of deployed robots, and area explored by each team. As can clearly be seen, we were able to deploy the largest robot team, while exploring an area bigger up to a magnitude than any other team. Due to the modest computational resources needed by the local approach, we were able to run 12 robots on a single *Pemtium4, 3GHz.*

Figure 3(a-b) depicts the joint trajectory of each team generated during the semi-final and final, whereas (c-d) shows the single trajectory of each robot of our team on the same map, respectively. The efficiency of the RFID-

TABLE I

RESULTS FROM ROBOCUP '06

| | | RRFr | GROK | IUB | SPQR | STEEL | UVA |
|---|---|---|---|---|---|---|---|
| PREL1 | # Robots | 12 | 1 | 6 | 4 | 6 | 1 |
| | Area [m2] | **902** | 31 | 70 | 197 | 353 | 46 |
| PREL2 | # Robots | 12 | 1 | 4 | 4 | 6 | 8 |
| | Area [m2] | **550** | 61 | 105 | 191 | 174 | 104 |
| PREL3 | # Robots | 10 | 1 | 5 | 7 | 6 | 7 |
| | Area [m2] | **310** | 59 | 164 | 44 | 124 | 120 |
| SEMI1 | # Robots | 8 | 1 | 6 | 4 | 6 | 6 |
| | Area [m2] | **579** | 27 | 227 | 96 | 134 | 262 |
| SEMI2 | # Robots | 8 | 1 | 6 | 5 | 6 | 7 |
| | Area [m2] | **1276** | 82 | 139 | 123 | 139 | 286 |
| FINAL1 | # Robots | 8 | - | 8 | - | - | - |
| | Area [m2] | **1203** | - | 210 | - | - | - |
| FINAL2 | # Robots | 8 | - | 6 | - | - | - |
| | Area [m2] | **350** | - | 136 | - | - | - |

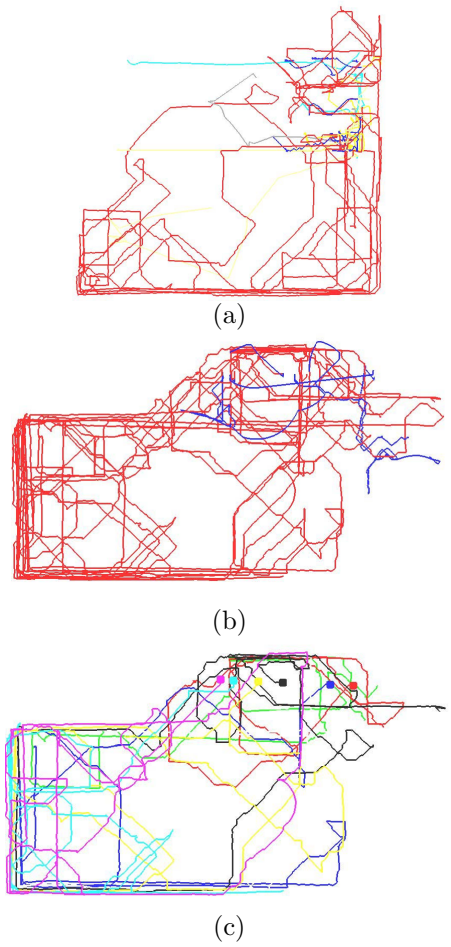

Fig. 3. Exploration trajectories recorded during the finals: (a,b) Comparison between our approach (red line) and all other teams. (c) Coordinated exploration of our robots, whereas each robot is represent by a different color.

based coordination is documented by the differently colored trajectories of each single robot.

### B. Evaluation of the global approach

Efficiency in terms of conflict detection and joint path length optimization has been evaluated on both artificially generated, and by a robot team generated RFID graphs. The artificially generated graphs, consisting of approx. 100 nodes, are weakly connected in order to increase the difficulty for the planning problem, whereas the graph generated by the robots, consisting of approx. 600 nodes, represents a structure naturally arising from an office-like environment.

Figure 4 depicts the result from evaluating greedy assignment, genetic optimized assignment, and sequence optimization on these graphs. Each method has been applied with a fixed number of randomized goals and starting positions, 10 times. We experimented different sizes of the robot teams, ranging from 2 to 20. The abrupt ending of the curves indicates the size of the agent team, at which no more solutions could be found, i.e. the scoring function returned infinity. Note that for all the experiments, the genetic algorithm was constrained to compute for no more than one second.

The result makes clear that sequence optimization helps to decrease both the overall path costs and the number of conflicts between single robot plans. Moreover, the method yields solutions with nearly no conflicts on the graph dynamically generated by the robot team (see Figure 4 (c)). In order to compare the global and local

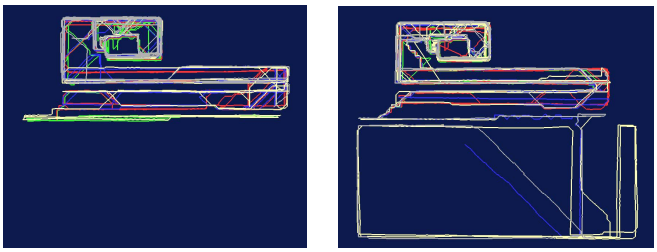

Fig. 5. Comparing the locally and globally coordinated exploration. During local exploration (a) robots get stuck in a local minima. The global approach (b) allows the robots to leave the local minima and to explore a larger area

approach in terms of the explored area, we conducted two experiments on a large map, for 40 minutes each (see [12] for a video). Due to the global approach, the robots were able to explore $2093m^2$ of the map, in contrast to the team executing the local approach, exploring only $1381m^2$ of the area. As can be seen by the trajectories in Figure 5, this was mainly because the robots running the local approach were not able to overcome the local minima in the long hall. With the global approach, the robots discovered the passage leading to the big area beneath the hall.

## VI. CONCLUSION

In this paper we presented a novel coordinated exploration mechanism for large teams of robots. This is basically composed of two parts. A first one, based on distributed local search, with the notable properties of not requiring direct communication and scaling with the number of agents. This approach can be seen as a stand alone system. A second, which monitors the local exploration restarting it in better locations. Both approaches are based on the use of RFIDs, which allow to build a significantly smaller representation of the environment compared to grid based approaches [13], [14],

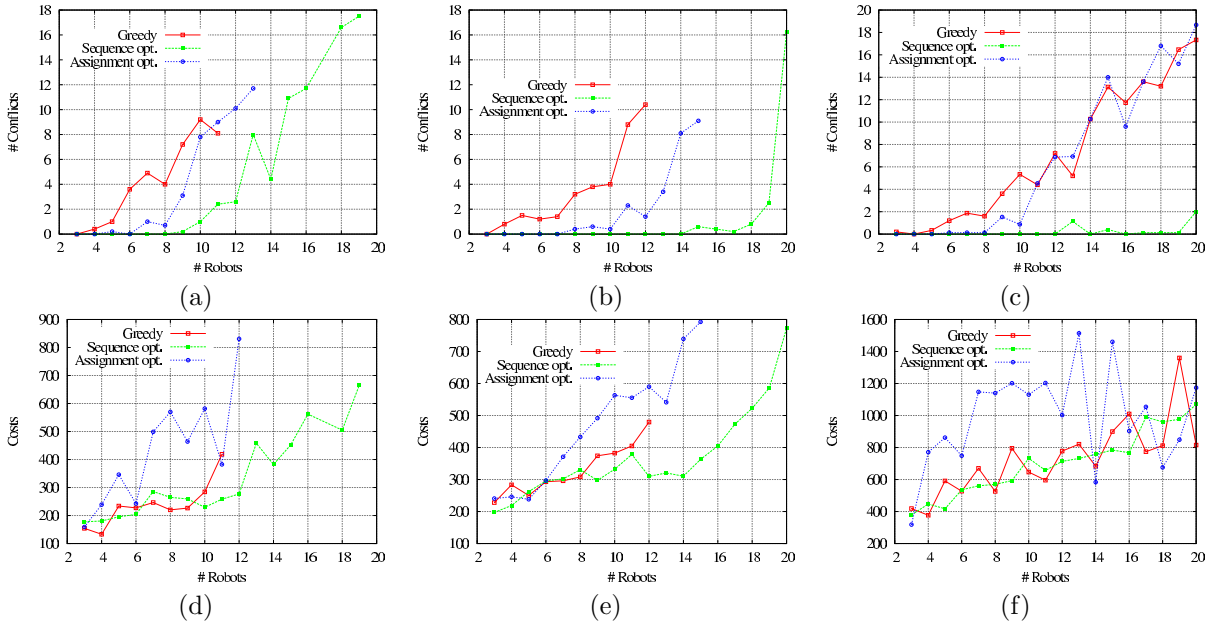


Fig. 4. Comparing the number of conflicts (a-c) and travel costs (d-f) of the three approaches on different RFID graphs: (a,d) narrow office-like environment, (b,e) narrow outdoor area, (c,f) graph generated from RFIDs deployed by the robots on a USARSim map.

[5], [4]. The experimental results from Robocup show that, for large environments, the local RFID approach definitively pays off. Moreover, for the global approach, we extensively experimented three approaches for solving the task assignment and planning problem. The first two are basically task assignment techniques [5] and the third, is a variant of sequential planning [4]. The experiments show that the sequential approach outperforms in most environments the assignment ones because of the capability of avoiding conflicts in the paths. Finally, qualitative experiments of the full system show that the method explored almost as much as double as the area explored by the local approach.

There are several issues we are planning to work on in the near future. Firstly, we want to run quantitative experiments for the full approach on USARSim and qualitative ones on the real robots. Furthermore, we are currently developing an new model for multi-robot plans based on Petri nets to improve the formal model of the Task Assignment and Planning problem [18]. This would allow us to model the plans as a discrete event system, where RFIDs as resources may be owned by one robot at the time. This would allow to take into account deadlock detection and plan time shifts (due to robots waiting) both at the planning and evaluation phases.

## References


[1] *Homepage of Robocup*, 2006. `http://www.robocup2006.org`.

[2] S. Balakirsky, C. Scrapper, S. Carpin, and M. Lewis. "USARSim: providing a framework for multi-robot performance evaluation". In *Proceedings of PerMIS 2006*, 2006.

[3] T. Balch and R. C. Arkin. Communication in reactive multiagent robotic systems. *Autonomous Robots*, 1(1):27–52, 1994.

[4] M. Bennewitz, W. Burgard, and S. Thrun. Optimizing schedules for prioritized path planning of multi-robot systems. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2001.

[5] W. Burgard, M. Moors, C. Stachniss, and F. Schneider. Coordinated multi-robot exploration. *IEEE Transactions on Robotics*, 21(3):376–378, 2005.

[6] S. Carpin, M. Lewis, J. Wang, S. Balakirsky, and C. Scrapper. Bridging the gap between simulation and reality in urban search and rescue. In *Robocup 2006: Robot Soccer World Cup X*. Springer, LNAI, 2006.

[7] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, December 1959.

[8] M. Erdmann and T. Lozano-Perez. On multiple moving objects. *Algorithmica*, (2):477–521, 1987.

[9] W. Hatzack and B. Nebel. Solving the operational traffic control problem. In *Proceedings of the 6th European Conference on Planning (ECP'01)*, 2001.

[10] M. Jaeger and B. Nebel. Decentralized collision avoidance, deadlock detection, and deadlock resolution for multiple mobile robots. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2001.

[11] A. Kleiner, J. Prediger, and B. Nebel. Rfid technology-based exploration and slam for search and rescue. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2006.

[12] A. Kleiner and V.A. Ziparo. Homepage of virtual rescuerobots freiburg. `http://gkiweb.informatik.uni-freiburg.de/~rescue/virtual/`, 2006.

[13] Yashodan Nevatia, Mentar Mahmudi, Stefan Markov, Ravi Rathnam, Todor Stoyanov, , and Stefano Carpin. Virtualiub: the 2006 iub virtual robots team. In *Proc. Int. RoboCup Symposium '06*, Bremen, Germany, 2006.

[14] Max Pfingsthorn, Bayu Slamet, Arnoud Visser, and Nikos Vlassis. Uva rescue team 2006 robocup rescue - simulation league. In *Proc. Int. RoboCup Symposium '06*, Bremen, Germany, 2006.

[15] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2003.

[16] J. Svennebring and S. Koenig. Building terrain-covering ant robots: A feasibility study. *Auton. Robots*, 16(3):313–332, 2004.

[17] B. Yamauchi. A frontier-based approach for autonomous exploration. In *IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA '97)*, 1997.

[18] V.A. Ziparo and L. Iocchi. Petri net plans. In *Proc. of ATPN/ACSD Fourth International Workshop on Modelling of Objects, Components, and Agents*, 2006.