Distributed Value Functions for MultiRobot exploration

Laëtitia Matignon, Laurent Jeanpierre, Abdel-Illah Mouaddib

Distributed Value Functions for Multi-Robot Exploration

Laëtitia Matignon and Laurent Jeanpierre and Abdel-Illah Mouaddib

Abstract— **This paper addresses the problem of exploring an unknown area with a team of autonomous robots using decentralized decision making techniques. The localization aspect is not considered and it is assumed the robots share their positions and have access to a map updated with all explored areas. A key problem is then the coordination of decentralized decision processes: each individual robot must choose appropriate exploration goals so that the team simultaneously explores different locations of the environment. We formalize this problem as a Decentralized Markov Decision Process (Dec-MDP) solved as a set of individual MDPs, where interactions between MDPs are considered in a distributed value function. Thus each robot computes locally a strategy that minimizes the interactions between the robots and maximizes the space coverage of the team. Our technique has been implemented and evaluated in real-world and simulated experiments.**

I. INTRODUCTION

The exploration consists of making decisions about where to go to gather knowledge about the environment. Exploration of unknown environments has been attracting considerable attention in autonomous mobile robotics given its wide range of applications including surveillance, search-and-rescue and planetary missions. In this context, multi-robot systems are often suggested to have obvious advantages over single-robot systems: faster, robust, fault-tolerant, compensation of sensors uncertainty, ... To minimize the overall time to complete the exploration task in the context of multi-robot exploration, efficient exploration techniques should consider strategies to distribute the robots over the environment to reduce the overlap between explored areas of each robot. This is the *global coordination* issue defined as allocating appropriately exploration goals for the individual robots so that they simultaneously explore different zones of the area. Favouring a well balanced spatial distribution of the robots in the environment has also the advantage to minimize the other main issue in multi-robot exploration, that is the interference between the members of the team. When robots are close to each other, *local coordination* such as collision avoidance is necessary. This can slow down the exploration because robots may spend time in these manoeuvres.

In this paper, we are interested in multi-robot exploration strategies that maximise the coverage and reduce the overlap between explored areas of each robot to efficiently explore by minimising local interactions. The context of this work is the CAROTTE challenge[1], where one or more robots have to map some unknown indoor area, to return within 30 minutes, to recognize and localize objects from a list of items that may be in the exploration area, and to do some opportunistic tasks like pushing a ball which position is unknown. We have at our disposal mobile robots able to communicate each others with their positions (when possible) and to construct local maps updated with the areas explored by the others. Throughout this paper, the localization aspect is not considered and we consider that robots are homogeneous, can share their positions and have access to such local maps. Under these assumptions, our objective is a distributed multi-robot exploration algorithm so that each robot constructs its exploration strategy locally by taking into account the others, their possible intended exploration target and other tasks that may be opportunistic at the same time. Additionally, strategies must be computed locally on-board by each robot with limited resources. Our approach must also be robust to communication breaks as communication may not be permanently available during the challenge.

Decentralized Markov Decision Process (Dec-MDP) provides an expressive means of modeling decentralized decision making problems but it is very hard to solve. Our approach is based on distributed value function (DVF) techniques. It decouples the multi-agent problem into a set of individual agent problems and considers possible interactions among team as a separate layer, which currently seems one of the main tracks to tackle scalability in Dec-(PO)MDPs [1], [2]. This could be seen as representing a Dec-MDP as a set of MDPs, one per robot, where interactions between MDPs are considered in the DVF. Our technique has been applied successfully in real-world scenarios during the CAROTTE challenge where our consortium was vice-champion.

This paper is organized as follows. Section II reviews the existing literature on the multi-robot exploration problem. Section III presents motivations and our approach based on DVF techniques. In section IV, the decision model of our framework is detailed. Finally, we show experiments in simulation and with real robots before concluding.

II. RELATED WORK

Several robot's exploration strategies have been proposed in the literature. If *a priori* information about the area to explore is available as a first coarse map, exploration goals can be a set of points of interest which must be visited to increase the map definition [3]. Otherwise, exploration goals must be found during the exploration at each time step. This technique is classically named *next best view*. It consists in selecting the next best exploration goal from the current set of candidates by choosing the one with the highest expected utility value; then computing and executing

The authors are with Université de Caen Basse-Normandie, CNRS - UMR6072 GREYC - F-14032 Caen France `firstname.lastname@unicaen.fr`

[1]http://www.defi-carotte.fr

the policy to reach this goal; lastly acquiring new data from the environment to update its map. To define appropriate exploration goals, a widely used approach is to extract *exploration frontiers* from the map. Exploration frontiers are defined as regions on the boundary between open space and unexplored space [4]. Then, a utility function captures in a value the interest of an exploration goal. In [4], [5], the utility function is the inverse of the cost of reaching an exploration frontier so the exploration strategy guides the robot to the closest unexplored area.

In the context of multiple robots exploration, each robot must be associated to its next best exploration goal. Most approaches assume that robots share the information they gathered so as to build a shared map and know their location and the relative ones of the others in this map. So as to allocate exploration frontiers to the individual robots, several exploration strategies have been proposed that mainly differ by the way global coordination is realised.

In [6], all the robots have the same exploration frontiers from the shared map and each robot is assigned to its closest frontier. This method does not require any central control but there is no global coordination as multiple robots can be assigned to the same frontier. In [7], [8], the utility of each target is computed as a compromise between the gain expected at this target (expected area discovered when the target will be reached taking into account the possible overlap in between robot sensors) and the cost for reaching this target. Global coordination is accomplished by assigning different targets to the robots, thus maximising the coverage and reducing the overlap between explored areas of each robot. In [9], the multi-robot exploration strategy uses a segmentation of the environment instead of a frontier-based approach to improve the space sweeping. In [7], [8], global coordination is centralized by a central agent that collects utilities from all the robots and assigns frontier-robot pairs, or that executes a target point selection algorithm assigning the robot-frontier pair with the best utility and recomputing the utilities given the new and all previous assignments. This process is repeated until all robots are assigned. Global coordination can also be decentralized, as in [10] where robots bid on targets thus negotiating their assignments. In [11], the utility function is for each robot-frontier pair the number of robots closer than it towards the considered frontier (in path distance). Thus, each robot is allocated to the frontier for which there is the less robots between the frontier and the robot to be assigned.

To summarize, most of previous work adopts either central agents or negotiation protocols with complicated process. Besides existing work in multi-robot applications does not address the local coordination issue as if a strategy that distributes well the robots over the environment would avoid all local interactions. In our CAROTTE challenge context, all the robots must start and return in a specified zone where close interactions will take place and then local coordination is necessary. Another limitation of these works, except for [8], is they do not address communication breaks issues.

III. DISTRIBUTED VALUE FUNCTIONS

Multi-robot coordination can be formalized as a decentralized decision making, *i.e.* each individual robot must find its policy so that all robots simultaneously explore different areas of the environment. In this section, we first motivate the choice of the Dec-MDP framework. Then background is summed up and our distributed value function method is presented.

A. Motivations

In single robot exploration, the exploration model is fully subsumed by the Partially Observable MDP (POMDP) framework. However, exploring using POMDPs meets the curse of dimensionality and limits its application to exploration domains because it considers uncertainty on observations, on outcomes of actions and on its own state. In our work, the localization aspect is not treated. We assume that the location of the robot is known at decision time. Therefore, we can simplify the POMDP model to an MDP as the explored environment is perfectly known and the robot has a known position, like we did in previous work on single-robot exploration [12]. Besides, MDP framework has many advantages: it allows for several goals at the same time and it accounts for uncertainty in the result of actions. MDPs can also be solved very efficiently through dynamic programming [13]. Finally, compared with most of published exploration algorithms using a two-steps planner [4], [7], [14], MDP planner is able to plan for its path at the same time as it chooses a location to explore or an opportunity task to do.

To consider the multi-agent issue, MDPs have been extended to Decentralized MDPs (Dec-MDPs). Dec-MDPs is an expressive framework for cooperative teams of decision makers but computing a solution to Dec-MDPs has a high complexity [15]. This typically limits the scalability in terms of number of agents and of the size of the area to explore. However, new coordination mechanisms for several robots have been developed recently that take advantage of local interactions and coordination to solve Dec-(PO)MDPs [**?**], [1], [2]. These methods represent a Dec-(PO)MDP as a set of interactive individual decision making problems and thus reduce its complexity. These interaction-based models for solving Dec-(PO)MDPs are based on the idea that an agent interacts only sometimes with some others. Thus the Dynamic Local Interaction Model (DyLIM) [2] splits the Dec-POMDP into two parts: the individual one "ignoring" the others and the coordination aspect describing how the agent influences the ones which it is in interaction with. These approaches are very interesting as they can compute solutions for larger problems while keeping the Dec-(PO)MDP formalism.

To address the multi-robot exploration problem, we use the Dec-MDP formalism solved as a set of individual MDPs, one per robot. To consider the interactions, we define a distributed value function computed locally by each robot that minimizes the interactions while maximizing the exploration. We will show that our model overcomes the limits of others since it can deal with a large number of robots; it

handles both the local and global coordinations; it is robust to communication breaks.

B. Background on Markov Decision Processes

Dec-MDP [15] is an extension of MDP for decentralized control domains. A Dec-MDP is defined with a tuple $< I, S, A, T, R, \Omega, O >$ and we have:

- I the number of robots, S the set of joint states and A the set of joint actions[2];
- $T : S \times A \times S \rightarrow [0, 1]$ a transition function; $T(s, a, s')$ is the probability for the I robots of transitioning from state s to s' after doing joint action a;
- $R : S \times A \rightarrow \Re$ a reward function mapping $S \times A$ to a real number that represents the robots' immediate reward for making joint action a while being in state s;
- Ω a set of observations and O an observation function can be left out if the states of agents are fully observable locally.

The goal of planning is to find a sequence of joint actions maximizing the long-term expected reward. Such a plan is called a policy π that is a mapping from S to A. An optimal policy π^* specifies for each state s the optimal joint action to execute at the current step assuming the agents will also act optimally at future time steps. The value of an optimal policy is defined by the optimal value function V^* that satisfies the Bellman optimality equation:

$$V^*(s) = \max_{a \in A}(R(s, a) + \gamma \sum_{s' \in S} T(s, a, s')V^*(s')) \quad (1)$$

where γ is the discount factor.

We can see an MDP [16] as a Dec-MDP where $I = 1$.

C. Distributed Value Functions for Multi-Robot Exploration

In our approach, we considered homogeneous robots, each of them having the same set of states and actions, and we solve the Dec-MDP as a set of individual MDPs: each robot constructs its exploration strategy locally while considering the others. A robot has access to a map updated with all explored areas and to the positions of the others. With these information, the robot can consider which areas could be explored by the other robots and thus reducing its interest in these areas. Formally, each robot computes its value function as the standard Bellman equation (eq. 1) augmented with this information.

Distributed value function (DVF) has been introduced in [17] as a way to distribute reinforcement learning knowledge through different agents. In [18], the distributed approach enables robots to learn multiple behaviours concurrently in a mobile robots navigation approach. Assuming that agent i knows at each step the state $s_j \in S$ of each other agent j, this leads to the following DVF applied to the exploration issue and from the viewpoint of agent i:

$$\forall s_i \in S \quad V_i(s_i) = R_{explo}(s_i) + \gamma \max_{a_i \in A} \sum_{s' \in S} T(s_i, a_i, s') \left[V_i(s') - \sum_{j \neq i} f_{ij} P_r(s'|s_j) V_j(s') \right] \quad (2)$$

This specifies that the expected gain of the robot i being in state s_i and going to s' is reduced by what the other robots can expect to gain by visiting the same state s'. The expected gain of another robot j is valued with the exploration expected gain at s', noted $V_j(s')$, weighted by the probability that robot j explores state s' from its current state s_j, noted $P_r(s'|s_j)$. Thus if a robot j has a high expected value for a state and if the probability for this robot to attain this state from its current one is great, then this robot may intend to go and explore the surroundings of this state. DVF tries to choose the action with a high expected gain for i so the one for which the interest of the others may be low. In other words, by locally computing DVFs and following resulting policies, the robots choose actions that minimize the interactions and that move the robots away from each other.

We now detail the different parts to calculate DVF. It requires to compute first data structures $< S, A, T, R_{explo} >$ of the model (§IV). The exploration reward function R_{explo} is computed with the reward propagation mechanism (§ IV-B.2). Then, the exploration expected gain of each other robots V_j is estimated. Given our communication restrictions, the robots cannot exchange information about their value functions. However each robot i can compute V_j by empathy. Additionally, since the robots we considered are homogeneous, this function is the same for each robot so this must be computed only once. This *empathic value function* is computed for each state $s \in S$ with the standard value iteration algorithm and the exploration reward function R_{explo}. To evaluate the probability $P_r(s'|s_j)$, an MDP_j for each other robot j is used. MDP_j has classical S, A, T structures of the model. But its reward function R_j is calculated for each MDP_j with rewards r_j located at the current position s_j of robot j. Using standard value iteration, the resulting value function V can be consistent with probability values: $V(s) = P_r(s|s_j)$. f_{ij} is a weighting factor that determines how strongly the expected gain of agent j reduces the one of agent i. It allows for balancing the value function with respect to the rewards, and is chosen as $maxR/maxV_j$.

Thus each robot computes strategies with DVF so as to minimize interactions. However it does not handle situations when the robots are close to each other that can happen for instance in the starting zone or in some narrow corridor. These situations with close interactions can be easily detected by computing the distance between the robots and its partners but require local coordination. Local coordination can be solved with a Multi-agent MDP (MMDP) [19] as the robots concerned by the interaction are close to each other. So joint policies are computed off-line for these specific joint states and followed when local coordination is detected.

[2]A state of the problem can be written with a tuple $s = (s_1, ..., s_I)$ such that s_j is the state of robot j. A_i defines the set of actions a_i of robot i.

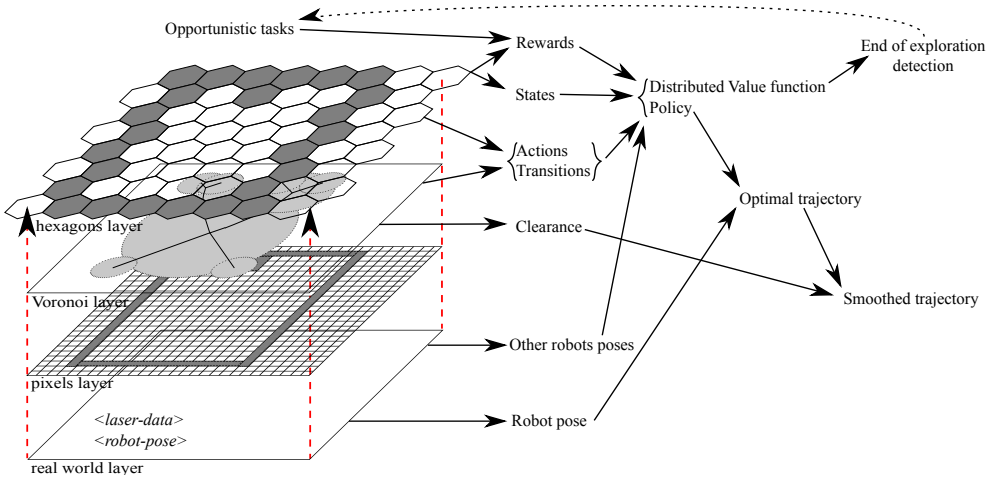


Fig. 1. The global software architecture of our decision framework for one robot.

IV. DESCRIPTION OF OUR MDP FRAMEWORK

The decision framework of our multi-robot exploration process is based on a set of MDPs, one per robot. Fig. 1 shows the global software architecture of our decision framework from the viewpoint of one robot. It is composed of a four layers grid from which are derived the data structures of one MDP model. A *decision step* consists in building the model, computing the policy from the distributed value function, and producing a smoothed trajectory. Classically, the planner stops after this and waits for the robot to arrive at the chosen location before starting again. Instead, we decide to allow the robot to plan continuously, updating its model and policy as it perceives changes in its environment. Indeed, continuous planning allows us to update quickly action plan if a target of opportunity is discovered or if the environment is dynamic. It also considers that information is gained *en route*: map is often explored before the robot reached its target. Another advantage is to react as soon as possible to the decisions of the others. However, this requires the decision step to be quick enough for online use. Given that the model will be updated at each decision step, we use the greedy approach that plans only for the immediate action. Following is the motivated description of the components of our architecture.

A. Four Layers Occupancy Grid

The first layer is the *real world layer* where the robots move. The second one is the *pixels layer* where raw sensor data are projected on the occupancy grid [20] using Bresenham's algorithm based on the current robot position and angle [12]. The occupancy grid represents the environment as a regular grid laid over the environment where each cell is initialized as unknown and updated as free (no obstacle) or occupied (something there) by the data acquisition process. The two last layers are computed from the occupancy grid and used to generate the data structures of the model. States and rewards are based on the *hexagonal grid*. This grid clarifies the pixel layer by merging nearby data. Each hexagon is composed of a set of cells and is considered as unknown, free or occupied according to the value of its cells. This grid has nice geometrical properties: it allows for 6 neighbours with the same distance from any cells that simplifies the reward propagation algorithm (§ IV-B.2) and increases the movement capabilities of the robot as compared to a square grid. For the actions and transitions, a *Voronoi diagram* [21] is generated.

B. MDP Model

1) State Space: The state space contains all possible locations of the robot and its orientation in the hexagonal grid (one location is an hexagon and it allows 6 orientations per location). An oriented location is referred as a *pose* in this paper. For a given exploration time, states are based on the current state of the hexagonal grid, including all the free hexagons.

2) The Reward Function: Since the objective is to explore unknown parts of the environment, the main reward component is based on the expected information gain in each pose. Various approaches for setting these rewards have been published [4], [7]. Similarly to [14], we compute a reward value for each hexagon. However, instead of characterizing the entropy of the cell itself, we estimate the information the robot would gain if it was at this location with some heading. To compute this value, we simply propagate rewards in some radius around frontier hexagons, respecting line-of-sight constraints [12]. Our actual system currently uses a grid with 15 centimetres hexagons, and propagates frontier rewards up to 2 meters. Targets of opportunity are simply added to the reward function as they happen, so that they are shared as well as exploration tasks.

3) Action Space: The robot actions model the movement of the robot: go forward, turn left or right, and wait. However, using only these actions and the hexagonal grid implies some troubles in cluttered space like a narrow corridor not oriented along one of the 6 hexagonal directions. In such a case the robot will have to zig-zag since it only goes from one hexagon to another. And if the corridor is narrow enough, this may prevent any traversal. To overcome this

situation, another robot action is added: follow the Voronoi edge to next hexagon. Please note that the Voronoi diagram is not sufficient for planning the whole trajectory as many tasks have to be done far from the Voronoi edges (starting positions, ball-to-push position). Additionally, trajectories following the Voronoi diagram frequently are suboptimal. Besides synchronizing several robots is easier when actions take the same amount of time: in a Voronoi diagram, some node can be connected to very close neighbours and to very distant ones at the same time.

4) The Transition Function: The transition function describes the expected outcomes of any action in any state. If the intended movement is allowed by the local safety clearance computed by the Voronoi diagram, the expected state will be reached with high probability. Otherwise, the state is not modified and a high penalty is added to the reward function for this state and this action. Allowed movements are not always deterministic because tight clearance may slow the robot down, or even make the expected move fail.

C. Smoothing the trajectory

The hexagonal optimal trajectory is smoothed, aggregating series of actions into way-points that do not violate safety clearances computed by the Voronoi Diagram. Figure 7 shows examples of smooth trajectories (the green ones) and hexagonal trajectories leading to jerky movements (the purple ones).

D. End of Exploration

End of exploration is detected simply by testing the expected value of the current best action for each robot. When it reaches zero, no more reward can be gained and we add a return reward to make the robot exit the area through its entry point.

V. EXPERIMENTAL PLATFORMS

Experimental results shown in this paper are from two main sources: experiments with two real robots and computer simulations with Player/Stage simulator.

A. Real Robots

The robot we use for exploration is a Wifibot[3] μ-trooper M. This 6-wheels mobile robot embeds an Intel Core 2 Duo processor, 2GB RAM and 4GB flash. It is also equipped with an Hokuyo URG-30LX[4] Laser range scanner. The software running on-board this robot is based on a Data Distribution System (DDS) implementation from OpenSplice[5]. This middle-ware allows for several programs to run concurrently, even on different computers. In our architecture, that implies that various modules can run asynchronously: Laser acquisition, SLAM, Decision, Mobility and Object recognition.

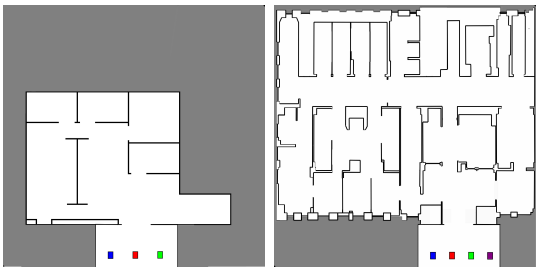
The architecture allows the robots to exchange their laser scans and their poses. Thus each robot knows the areas explored by the others and updates its local map with local and distant scans. In particular, the *SLAM module* receives laser readings and provides the other modules and other robot with the robot pose (location and heading). It is based on [22]. The *mobility module* implements an advanced point and shoot algorithm, along with a backtrack feature that prevents the robot from being stuck, reverting back on its own trajectory. The point and shoot algorithm consists of turning to the desired heading and then going forward the specified distance. Here, the algorithm accepts a small initial angular error, and keeps correcting the heading drift as the robot goes forward. The *decision module* runs asynchronously, computing a new policy every second in average.

B. Simulated Robots

We use the Player/Stage[6] [23] simulator with an architecture that mimics the real robots. DDS is replaced by an IPC (Inter Process Communication) shared memory segment. Laser acquisition is simulated by a "laser" virtual sensor tuned with the URG-30 properties. A "position" virtual device simulates both the SLAM module by providing odometric data and the mobility module by executing the point and shoot algorithm. Finally the decision module used on real robots can be used with the simulator without modification.

VI. EXPERIMENTAL RESULTS

A. Simulated Robots



(a) autolab. (b) hospital.

Fig. 2. Simulation environments from Player/Stage with starting poses.

We varied the number of robots from one to five and used two different simulated environments from Player/Stage (Fig. 2). Robots are initially in a line formation in the starting zone and to reduce the situations of local coordination at the beginning, each robot starts with 15 seconds of delay from its precedent. We plot how the coverage of the environment to explore evolves over time, while varying the number of robots. Fig. 3 and 4 report the time it takes to cover 50, 70, 90, 95 and 100% of the environment. The *EndM* coverage corresponds to the end of the mission, *i.e.* when all the robots have returned to their starting poses after they detected the end of exploration.

It is of interest to notice two stages in the coverage evolution: the *beginning stage* until 95% and the *end stage* consisting of 100% and the end of the mission. During the beginning stage, robots spread out to different areas and

[3]www.wifibot.com
[4]www.hokuyo-aut.jp
[5]http://www.opensplice.com
[6]http://playerstage.sourceforge.net/

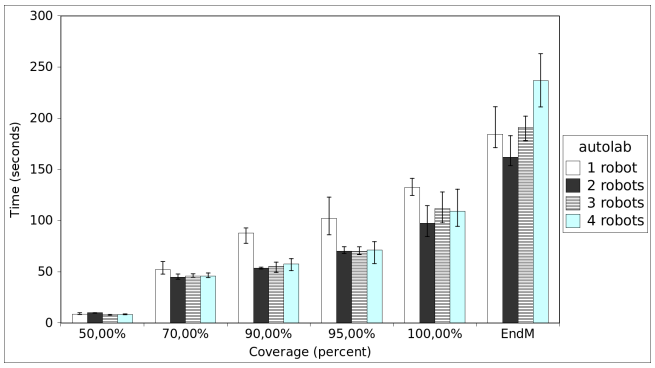


Fig. 3. Results from autolab environment (averaged over 5 simulations).

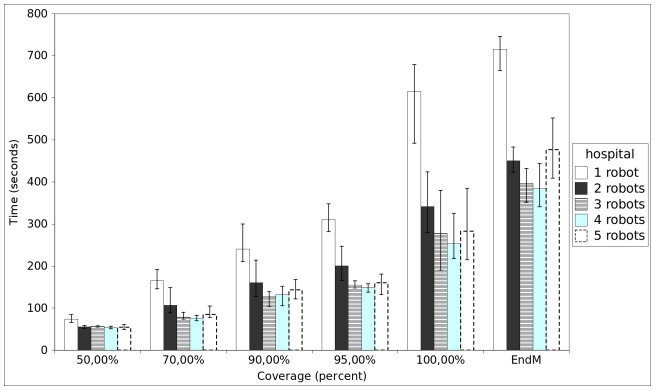
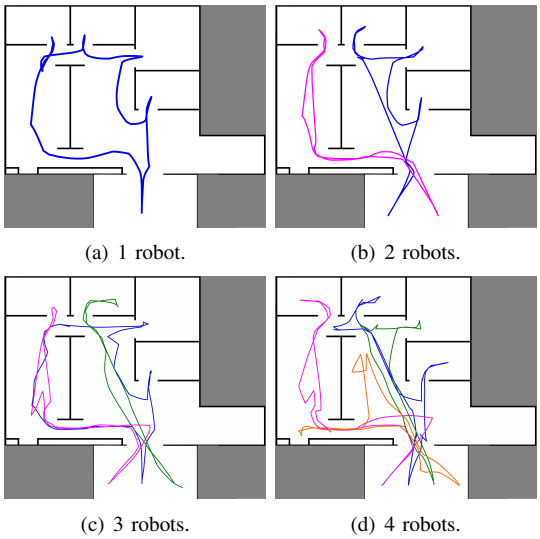


Fig. 4. Results from hospital environment (averaged over 5 simulations).



(a) 1 robot. (b) 2 robots.

(c) 3 robots. (d) 4 robots.

Fig. 5. Paths taken by the robots in the autolab environment.

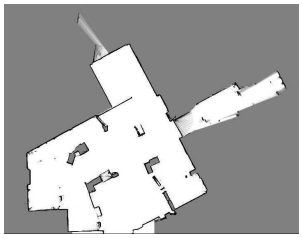


Fig. 6. Resulting pixel map of an area explored with two robots. Pixels colour ranges from black (obstacle) to white (free). Gray shades represent the uncertainty.

covered the space efficiently, as illustrated by paths followed by 2 and 3 robots in Fig. 5. However, there is a number of robots beyond which there is not much gain in the coverage and that depends on the structure of the environment [7]. For instance in the autolab environment (Fig. 3), even if two robots perform significantly better than one robot, there is not much gain in having three or four robots. Indeed, two robots can share separate zones (Fig. 5b) but the gain of having more robots is low compared to the overlap in trajectories and the risk of local interactions ((Fig. 5c and d). In the hospital environment (Fig. 4), four robots seems to be the optimal number. Besides, beyond this number of robots, the end stage takes longer as robots end up interfering with one another to cover few last spots or to return to start. Thus during the end stage, local coordination cannot be avoided and the more robots are used, the longer the end stage may take as local interactions require bypasses, backtracks and stops to let pass.

B. Real Robots

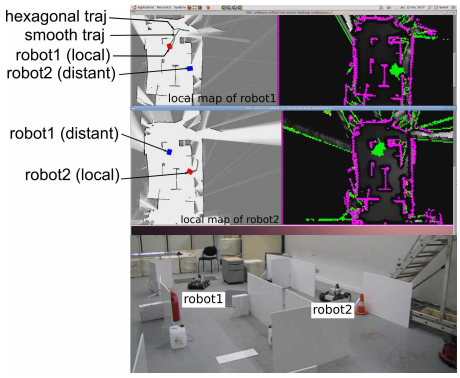
We also performed experiments with our two μ-troopers. The videos accompanying this paper[7] show the exploration of the robots and some situations of local coordination successfully resolved. One resulting map of the area obtained at the end of a mission is in Fig 6.

[7]Also available at http://lmatigno.perso.info.unicaen.fr/research

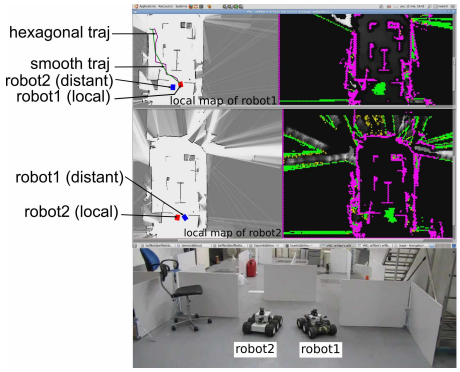
In Fig. 7, the bottom part is the real-world situation while the top part shows the decision model with local maps on the left and value functions on the right. Each local map is shown as seen by each robot in which the distant robot is seen by the local one. In Fig. 7a, we observe that robots share the space to better explore the space. Robot 2 (bottom part of the figure) decides to explore a corridor while the other (the top part of the figure) decides to explore the top of the map. In the right of both maps, we show the value function where the gray cells have some rewards, the black ones are with no reward and the green one show the space surrounding the other robot in the map. Fig. 7b depicts a local coordination situation. We observe that robot 1 decides to move while robot 2 waits for the other to pass the door. No breaks in communication occurred during these experiments. However, we observed some temporary network errors during other tests and we noticed that our DVF approach resulted in an efficient exploration process. Once the connection had been re-established, the robots coordinated with each other again as our architecture is robust to communication failures.

VII. CONCLUSIONS AND FUTURE WORKS

In this paper, we address the problem of multi-robot exploration with a fully decentralized approach based on distributed value functions. Our approach considers a Decentralized MDP as a set of individual MDPs where in-



(a)



(b)

Fig. 7. a) Global coordination and consistent local maps. b) Local coordination.

teractions between MDPs are considered in the distributed value function. Thus each robot computes locally a strategy that minimizes the interactions between the robots and maximizes the coverage of the team. Local coordination is also considered so as to resolve situations with close interactions. Experimental results show that this method is able to effectively coordinate a team of robots during exploration.

One interesting research direction and a planned restriction of the challenge is to consider communication loss constraints during the exploration. Our decentralized approach does not rely on perfect communication, but at reduced efficiency. In case the robots do not know at each time step their relative poses, our DVF approach can still consider the intentions of others from their last known poses. If in addition local maps are not updated, each robot will explore all targets. We want to investigate communication breaks in more detail and to test our DVF approach extended to situations where communication between robots drops out, as introduced in our previous work [24]. Besides, our system currently takes "randomly" pictures for the object recognition, *i.e.* it takes pictures at a specified rate along the trajectory planned for the exploration. Another perspective is to plan to perceive. Indeed robots can act to improve object detections at the same time they explore. We would like our robots to modify their strategies to both maximise the exploration and minimize false detections.

VIII. ACKNOWLEDGEMENTS

This work has been supported by the NRA (French National Research Agency) and the DGA (Defense Procurement Agency) (ANR-09-CORD-103) and jointly developed with the members of the Robots_Malins consortium.

REFERENCES

[1] A.-I. Mouaddib, M. Boussard, and M. Bouzid, "Towards a formal framework for multi-objective multiagent planning," in *Proc. of AAMAS*, 2007.

[2] A. Canu and A.-I. Mouaddib, "Collective decision-theoretic planning for planet exploration," in *Proc. of ICTAI*, 2011.

[3] G. Lozenguez, L. Adouane, A. Beynier, P. Martinet, and A.-I. Mouaddib, "Map partitioning to approximate an exploration strategy in mobile robotics," in *Proc. of Int. Conf. on Practical Applications of Agents and Multi-Agent Systems (PAAMS)*, 2011, pp. 63–72.

[4] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *In Proceedings of the IEEE International Symposium on Computational Intelligence, Robotics and Automation*, 1997, pp. 146–151.

[5] S. Koenig, C. Tovey, and W. Halliburton, "Greedy mapping of terrain," in *Proceedings of ICRA*, 2001, pp. 3594–3599.

[6] B. Yamauchi, "Frontier-based exploration using multiple robots," in *Proceedings of the second international conference on Autonomous agents*, ser. AGENTS '98, 1998, pp. 47–53.

[7] R. Simmons, D. Apfelbaum, W. Burgard, D. an Moors M. Fox, S. Thrun, and H. Younes, "Coordination for multi-robot exploration and mapping," in *Proc. of the AAAI NCAI*, 2000.

[8] W. Burgard, M. Moors, C. Stachniss, and F. Schneider, "Coordinated multi-robot exploration," *IEEE Transactions on Robotics*, vol. 21, pp. 376–386, 2005.

[9] K. M. Wurm, C. Stachniss, and W. Burgard, "Coordinated multi-robot exploration using a segmentation of the environment," in *Proceedings of IROS*, 2008, pp. 1160–1165.

[10] R. Zlot, A. Stentz, M. Dias, and S. Thayer, "Multi-robot exploration controlled by a market economy," in *Proc. of ICRA*, vol. 3, 2002, pp. 3016–3023.

[11] A. Bautin, O. Simonin, and F. Charpillet, "Towards a communication free coordination for multi-robot exploration," in *6th National Conference on Control Architectures of Robots*, 2011.

[12] S. L. Gloannec, L. Jeanpierre, and A.-I. Mouaddib, "Unknown area exploration with an autonomous robot using markov decision processes," in *Proc. of Towards Autonomous RObotic Systems*, 2010, pp. 119–125.

[13] R. Bellman, *Dynamic programming: Markov decision process*, 1957.

[14] C. Stachniss and W. Burgard, "Exploring unknown environments with mobile robots using coverage maps," in *Proc. of the Int. Conf. on Artificial Intelligence (IJCAI)*, 2003, pp. 1127–1132.

[15] D. S. Bernstein, R. Givan, N. Immerman, and S. Zilberstein, "The complexity of decentralized control of markov decision processes," *Math. Oper. Res.*, vol. 27, pp. 819–840, 2002.

[16] M. L. Puterman, *Markov decision processes*. John Wiley and Sons, 1994.

[17] J. Schneider, W.-K. Wong, A. Moore, and M. Riedmiller, "Distributed value functions," in *In Proc. of ICML*, 1999, pp. 371–378.

[18] S. Babvey, O. Momtahan, and M. R. Meybodi, "Multi mobile robot navigation using distributed value function reinforcement learning," in *Proceedings of ICRA*, 2003, pp. 957–962.

[19] C. Boutilier, "Planning, learning and coordination in multiagent decision processes." in *TARK*, 1996.

[20] A. Elfes, "Using occupancy grids for mobile robot perception and navigation," *Computer*, vol. 22, pp. 46–57, 1989.

[21] F. Aurenhammer, "Voronoi diagrams – a survey of a fundamental geometric data structure," *ACM Computing Surveys*, vol. 23, no. 3, pp. 345–405, 1991.

[22] J. Xie, F. Nashashibi, N. M. Parent, and O. Garcia-Favrot, "A real-time robust slam for large-scale outdoor environments," in *17th ITS World Congress*, Oct. 2010.

[23] B. P. Gerkey, R. T. Vaughan, and A. Howard, "The player/stage project: Tools for multi-robot and distributed sensor systems," in *In Proc. of the Int. Conf. on Advanced Robotics*, 2003, pp. 317–323.

[24] L. Matignon, L. Jeanpierre, and A.-I. Mouaddib, "Distributed value functions for multi-robot exploration: a position paper," in *AAMAS Workshop19: Multiagent Sequential Decision Making in Uncertain Domains (MDSM)*, 2011.