Balanced Multi-Robot Exploration through a Global Optimization Strategy

Ling Wu, Domenec Puig and Miguel Angel Garcia

Abstract— **This paper reviews the state of the art in coordinated multi-robot exploration and proposes a new exploration objective based on a practical scenery, reducing the difference of waiting time among different regions of a workspace, which has not been still considered in the literature. A new global optimization strategy for coordinated multi-robot exploration based on a proper dispersion of robots in separate regions is presented. This strategy aims at achieving the lowest variance of regional waiting time and the lowest variance of regional exploration percentage. Both features reveal that the proposed strategy performs better than other state of the art approaches.**

Index Terms— **Multi-robot exploration, multi-robot coordination, waiting time variance, K-Means.**

I. Introduction

The exploration of unknown areas by means of mobile robots is an important issue that has been widely studied along the last years. Its importance lies in the large number of applications that require robotic exploration, such as search and rescue, planetary exploration, or reconnaissance, among others.

Coordination has been considered in the exploration problem when teams of robots are to be deployed. Multiple robots can carry out tasks faster than a single robot. Nonetheless, a multi-robot system is not n-folded productive due to a variety of factors. First of all, space limitations in the environment can force the robots to move together. Second, multiple robots can interfere with each other [1]. Third, without knowing the existence of the other robots or their locations, a robot could explore the same places that other robots have already searched. Thus, coordination is necessary to increase the system gain in a multi-robot system.

This paper describes a new K-Means based (KME) global optimization strategy for coordinated multi-robot exploration that builds on a previous proposal [2]. The main objective consists of reducing the difference of waiting time among different regions of a workspace. The proposed method is compared to three representative exploration algorithms: Yamauchi [3], Burgard et al. [4], [5], and market economy by Zlot et al. [6]. A statistical study presented in this paper reveals the greedy behavior of those proposals, in the sense that they can explore some parts of the environment much later than others. In contrast, the proposed planner ensures a balanced exploration of the environment by forcing the explicit dispersion of robots over it. This behavior is desirable in a variety of applications, such as search and rescue.

This article is organized as follows. Section II reviews the state of the art on multi-robot exploration. Section III proposes a new team objective for exploration applications and the need for a global optimization strategy. Section IV describes the proposed multi-robot coordinated exploration algorithm. Section V presents the experimental setup, the definition of two new features (the regional waiting time variance and the regional exploration percentage variance) and the experimental comparison among the tested approaches in terms of both features. Finally, conclusions and further improvements are given in Section VI.

II. State Of The Art

This section summarizes different well-known coordinated multi-robot exploration methods. The benefits and defects of those methods are also discussed.

The algorithm proposed by Yamauchi [3] is the simplest exploration strategy. Each robot simply looks for a frontier cell that can be reached with the lowest cost. It is a greedy strategy with no coordination among robots.

The algorithm by Burgard et al. [4], [5] aims at reducing the exploration completion time and thus coordinates the robots to explore as much area as possible. In order to achieve this goal, a decision-theoretic approach trades off the *utility* and the *cost* of visiting targets. The cost of a target is the length of the optimal path that a robot takes to visit it, whereas the utility of a target is the area expected to be found when the robot arrives at it. The utility of a target is reduced if it is inside the sensor range of assigned targets of other robots. Therefore, robots will tend to disperse. However, the degree of dispersion is local, that is, they disperse until the overlap of the expected new exploration areas by different robots is zero. This does

Ling Wu, Domenec Puig are with the Dep. of Computer Science and Mathematics, University Rovira i Virgili, Av. Dels Paisos Catalans, 26, 43007 Tarragona, Spain. E-mail: {ling.wu,domenec.puig}@urv.cat

Miguel Angel Garcia is with the Department of Informatics Engineering, Autonomous University of Madrid, Francisco Tomas y Valiente 11, 28049 Madrid, Spain, E-mail: miguelangel.garcia@uam.es

This work has been partially supported by the Spanish Ministry of Education and Science (MEC) under project DPI2007-66556-C03-03. Ling Wu is supported by a FPI scholarship from the Spanish MEC.

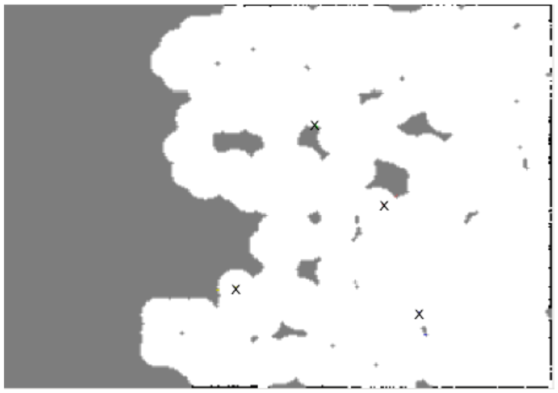

Fig. 1. Example of the greedy behavior of [6]. Four robots (marked with X) start exploring from the bottom right corner (the workspace is a blank area without any obstacles). After quite a number of exploration steps, no robot has arrived at the left part of the workspace. The gray and white areas correspond to unexplored and explored free space respectively.

not guarantee dispersion over the whole workspace. Therefore, robots behave similarly to those of Yamauchi's method [3]. The method does not apply any optimization algorithm to organize the assignment of targets to robots.

The algorithm by Fox et al. [7] deals at the same time with exploration and mapping with multiple robots that do not know their relative positions. The positions of frontiers and hypothetical positions of other robots are considered as targets. The definition of cost and utility for frontiers is similar to [4], [5]. It organizes the robot-target assignment by an optimization scheme, which maximizes the total benefit in each decision step. However, it does not force the dispersion of robots. Therefore, robots can go to different targets that are close to each other. The behavior of robots is similar to [3] as well.

Simmons et al. [8] outperformed [4], [5] in two ways. Firstly, they improved the estimation of the *information gain* that robots are supposed to attain when they arrive at their targets. The *information gain* is the amount of area that a robot is expected to discover at the position of a target, subtracting the overlapped areas expected to be found by other robots. Therefore, it forces robots to disperse as [4], [5]. Secondly, this method proposes the concept of bids. Bids are submitted by individual robots, which calculate them as the expected information gain minus the cost of visiting the goals, similar to the trade-off (benefit) value of [4], [5]. A center agent collects the bids from all robots and assigns the target to the robot which offers the highest bid. This method does not optimize the overall assignment performance.

The approach by Zlot et al. [6], a well-known market-economy based method, is a further improvement to [8]. In [8], robots choose targets from frontiers during the exploration process. Those frontiers lie at the border of the free known space. Instead, in [6], each robot selects some targets from the unexplored workspace so that they form a representative subset of the whole set of future unexplored goals. Targets are auctioned by robots (auctioneers) to others robots (bidders) with proposed prices. Bidders calculate their bids as the expected benefit, which is the expected *revenue* (utility) minus the expected cost. If the bid calculated by a bidder is lower than the proposed price, the bid will not be submitted. If the bids sent by bidders are higher than the proposed price, the auctioneer will choose the bidder with the highest bid. If no bid is submitted for the auctioned job, the auctioneer will keep the job for itself. In short, who offers the highest price for a target will finally win it. If a robot wins the auction of a target, it adds this target to its route list for visiting it in the future.

The auction algorithm is actually an optimization solution to a classical assignment problem in which there are *N* persons and *N* tasks, and each task *j* has a respective benefit value $b_{i,j}$ if it is assigned to person *i,* and the objective of the problem is defined as finding out the maximized total benefit of an assignment. Bertsekas [9] showed that his auction algorithm can effectively find an optimal solution that maximizes the total benefit. Therefore, the market economy based algorithm is an optimization assignment solution for assigning robots to tasks in each decision step. It is similar to the proposal in [7] or any other that applies an optimal solution for maximizing the total benefit for *N* pairs of robot-task assignments.

Although the market economy method has applied the above described optimization of total benefit (*utility - cost*), and it has tried to improve the global productivity by considering the future targets, it has three problems. Firstly, the method does not explicitly force the robots to disperse. The optimization is undermined similarly to the case of [7]. Secondly, some robots will obtain more targets since they are closer to them (their costs are lower than those of the robots which are farther away from their goals). Consequently, they are always closer to the farther targets in the same direction since they have arrived before other robots and, therefore, the workload is not balanced among the robots in most of times. Thirdly, targets are auctioned again in future decision steps in order to further improve the decision when robots change their positions. However, targets which previously belong to a robot at earlier steps will probably be acquired by another robot if the latter is currently closer to them. Therefore, this approach will finally behave like the greedy strategy [3], in which robots always choose the closest targets. This will be further explained in section III.

The exploration algorithm based on K-Means clustering proposed in [2] has overcome the greedy characteristic of the previous strategies by sending robots to different regions of the workspace. Unknown cells of an occupancy grid are clustered into as many disjoint regions as available robots by applying the K-Means clustering algorithm, with *K* being the number of robots. Each robot is then assigned to its closest region according to the Euclidean distance from the robot to the region's centroid. Each robot is assigned the frontier cell with the lowest cost, which is calculated by summing: (a) the length of the shortest path between the robot and the frontier cell, (b) a penalization of the Euclidean distance from the frontier cell to the centroid of the region assigned to that robot, provided the frontier does not belong to that region, (c) a constant penalization in case the frontier cell is within the sensor range of another frontier cell assigned to a different robot (this guarantees the repulsion between robots). All robots start moving to their assigned frontier cells until the first robot reaches its destination. At that point, the target assignment will be decided again.

However, experiments show that some flaws undermine the overall exploration efficiency of that algorithm. Firstly, the assignment of robots to regions can be improved by an optimization strategy instead of by assigning robots to regions on a sequential basis. The latter has been found to cause a longer total sum of routes traveled by robots. Secondly, the estimation of geometric distances between robots and centroids of regions (used to decide the assignment of robots to regions) is a very coarse estimation of the real distance between robots and regions, and it does not fully exploit the actual path length that may be already known. For example, when a big obstacle has already been found between a robot and a region, the shortest route length from the robot to that region through free space is the actual distance between them. If the distance between robots and regions is more accurately defined, the assignment of robots to regions will be more efficient.

Finally, the penalization of the distance from a frontier to the centroid of a region when a robot does not choose a target frontier beside its own region, may cause the robot strongly favors frontiers beside its own regions. Therefore, it will tend to ignore unexplored cells of the regions assigned to other robots that are on its way to its own region. Moreover, since the process repetitively assigns robots to regions during exploration, a robot usually changes its target according to its assigned new region. When a robot is continuously assigned to different regions and since it favors the frontiers of its own region, it can change its target drastically and move back and forth without exploring new area. Hence, the target selection should be stabilized in order to prevent that undesirable behavior. The algorithm proposed in this paper, which is described in section IV, improves the aforementioned aspects.

III. Exploration Objective

In exploration problems, it is not possible to know the optimal routes for the robots until the map of the workspace is revealed. Hence, an approach that makes an optimal choice in every decision step does not necessarily lead to a global optimal solution. When an optimal solution is intractable (since the problem is NP-hard), it is interesting to achieve a close to optimal solution. In order to do that, it is necessary to apply a global strategy to drive the exploration process according to a certain objective. A hierarchy of decision planners should be considered: a planner to achieve the global exploration objective and the planners to refine every decision step. We propose to apply three levels of optimization in a coordinated multi-robot exploration algorithm: (a) path planning strategy; (b) task assignment optimization strategy at every decision step; (c) global optimization strategy in order to overcome the greedy behavior caused by the second level optimization strategy.

The path planning strategy is the lowest level optimization strategy of the exploration algorithm. It decides the path that a robot will follow to arrive at a chosen target. In order to avoid obstacles when robots are traveling, an obstacle-free path should be planned. Furthermore, even if the main objective of the exploration application is not to minimize the total traveling cost of robots, minimizing that cost is still a basic requirement in order to guide robots to travel more efficiently. For example, the approaches presented in Section II, which aimed at minimizing completion time, have applied some kind of optimal route calculation.

Most of the state of the art strategies mainly focus on how to assign tasks to robots. Some of them (such as [6],[7]) applied a second level optimization algorithm. The second level optimization aims at efficiently assigning the robots to the next targets in every decision step. With the information obtained at each moment, although there are uncertainties about the contents of the workspace, the decision maker will try to do its best to assign the targets to the robots in order to minimize the completion time or the total traveling cost. Combinatorial methods can be used in order to make an optimal solution. If the optimal solution cannot be found in polynomial time, a heuristic solution can be used instead.

The highest level optimization aims at compensating the greedy behavior caused by the second level optimization. Without a global view, any optimization algorithm can end up yielding a local optimum. In [3]-[6], each robot tends to move to a target as fast as possible in order to minimize the completion time. Therefore, they all choose the closest targets except that in [3], [4] where dispersion is considered but only at a local level. Robots in the market economy method [6] show similar greedy behavior —exploring some parts of the workspace much later than others (see Fig. 1).

A. *Minimizing the Waiting Time Variance for Each Region of Workspace*

Based on the above motivation, we propose a new objective function that can be useful for many exploration applications. It consists of minimizing the variance of waiting time of every region in the working space (MINIWTV). Minimizing the waiting time variance (WTV) is an important problem in scheduling [10]. An analogue of this proposition is to minimize the waiting time variance of services in computer networks. When the clients' WTV of network services is minimized, that is, they get all the services after waiting for similar time slices, clients feel that the quality of service (QoS) of the network services is stable and the performance of the network is predictable. In search and rescue applications, if the different regions of the working space can be explored with similar arrival times (minimizing the waiting time variance of different regions), potential victims in a region will not have to wait much longer than those in other regions.

The algorithm proposed in the present paper (KME) aims to achieve the above objective. First, it partitions the unexplored space into as many areas as available robots by applying the technique first introduced in [2]. This partitioning acts as a global optimization strategy. Once the unknown space is divided into regions, each region is assigned to a robot by a Linear Programming (LP) algorithm in order to minimize the total path cost, considering the relative location of robots and regions. Section IV.B describes how to define the distance from a robot to a region. Section IV.C explains how to decide the next target a robot should visit. These processes act as a second level optimization strategy. Hence, the KME method is a globally optimized approach that reduces the waiting time in

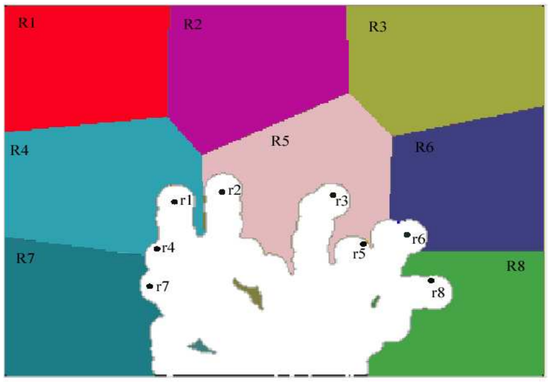

Fig. 2. Example of K-Means clustering algorithm used to partition an empty working space in the proposed multi-robot exploration method. A partition of unknown space into eight regions and robots ($K = 8$) is shown. Every region is assigned to a single robot. The white area represents space that has already been explored, while the colored regions are the clusters produced by K-Means.

different regions, since robots will disperse to explore different unknown areas partitioned by the K-Means clustering algorithm, avoiding thus the greedy behavior of previous methods, which can partially explore a part of the map long after another.

IV. Coordinated Exploration Algorithm Based on K-Means

The proposed multi-robot coordinated exploration algorithm has several advantageous features comparing to the state of the art approaches. Firstly, robots disperse globally so as to reduce the difference of the waiting times between different regions of the workspace. Secondly, at every moment, the workload distributed to a group of robots is balanced. The applied K-Means clustering divides the exploration tasks into balanced target sets for robots. This favors that different parts of the workspace be explored at a similar speed. Thirdly, exploration is coordinated so as to efficiently reduce the completion time. The optimal assignment of regions to robots is formulated as a Linear Programming (LP) problem and solved by an LP solver [11] so that the total route traveled by the robots is the shortest. This reduces the overall completion time. Furthermore, the dispersion of robots allows them to explore the unknown space in parallel. This also reduces the total route that robots must travel during exploration. When robots are not inside their assigned regions, they will explore the targets that lie on the way to their own regions. When they have arrived at their own regions, the exploration mechanism is similar to that of Burgard et al. [4], [5]. Therefore, it reduces also the completion time of the exploration process. Fourthly, global dispersion and workload balancing are

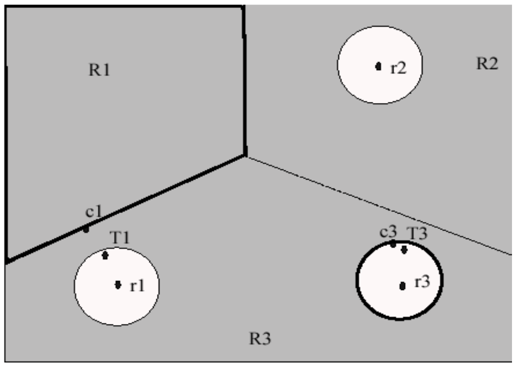

Fig. 3. Gray represents unexplored space that has been partitioned into regions R1, R2, and R3 (three robots r1, r2 and r3 are considered). White represents free spaces that have already been explored by r1, r2 and r3. R2 is accessible for r2. R3 is accessible for r1, r3. R1 is inaccessible for r1, r2 and r3. R1's contour line has been marked with a dark line. For r3, its accessible contour cells of R3, a circle, are marked with a dark line.

dynamically maintained during exploration. This leads to a proper assignment of robots to regions and targets to robots, increasing in this way the overall efficiency of the exploration.

This section is organized as follows. Section IV.A describes the partitioning method. Section IV.B and IV.C present the strategies for optimizing the robot-region and robot-target assignments. The overall exploration algorithm is summarized in Section IV.D.

A. *Partition of the Unknown Space through K-Means*

In the proposed algorithm, the unknown workspace is partitioned into as many regions as robots by means of the same K-Means based technique first proposed in [2] (see Fig.2). In particular, all unknown cells in the map's occupancy grid are clustered by applying the well-known K-Means algorithm. The iterative process is described in [2], [12]. Since the unexplored space is partitioned into regions and they will be explored by different robots in parallel, the workload is distributed among robots in a balanced way and the waiting time of each region in the working space is reduced. Thus, for search and rescue applications, victims dispersed over the working space will be found by robots that are in their vicinities more quickly. By working simultaneously in separate parts of a map, robots keep a reasonable dispersion among them. In other solutions that do not apply such a global strategy (e.g., [3]-[6]), the robots may well concentrate in some parts of the workspace, reaching other parts significantly later (see Fig.1).

B. *Robot-Region Assignment*

After the unknown space is divided into K regions, the algorithm decides which region is assigned to each robot (see Fig.2 and Fig.3). The proposed algorithm distinguishes between two types of regions and defines the distance for each case. In the first case, when a region is *accessible* (*RA*) for the robot, it can move straight ahead to a free cell which lies at the verge of this region. The route is calculated by a method like the one proposed in [4], [5]. This algorithm will automatically avoid collision with walls and have the shortest cost, which will be referred to as real path cost. The distance from a robot to this region is defined according to this cost.

In the second case, if there is no free space lying between a robot and a region, the robot will first explore other unknown regions before reaching its assigned region, therefore, the region is *inaccessible* (*RI*) for it. In Fig. 2, for example, the upper three regions are inaccessible for all robots. The distance from a robot to an inaccessible region is a geometric distance estimated as described next.

First of all we define the distance between a robot r and a contour cell c of a region R. A contour line is the set of unexplored cells that lie on the edge of R. A frontier cell f is an explored free cell that lies beside a contour line. Robots can go to a frontier cell and use their sensors to explore a new area. If a contour cell c has free neighboring frontier cells F_c such that r can go directly to c through free space, then c is accessible for r. If R has accessible contour cells for r, R is accessible for r (note that even if a contour cell is beside free space, but r is not in that space, R is still not accessible for r). The distance between r and an accessible contour cell c_a of RA is defined as $d(r, c_a)$, the shortest distance from r to all possible frontier cells adjacent to c.

$$d(r,c_a) = \min\{\delta(r,f) \mid f \in F_c\} \qquad c_a \in RA \tag{1}$$

where $\delta(r,f)$ is the real path cost from r to a cell within F_c, when c is accessible for r.

When a contour cell is in an inaccessible region R for r (it is denoted as c_i), since r can not arrive at it immediately, an estimation of distance is defined as the geometric distance between r and c. If there is an obstacle on the direct line connecting r and c_i, the distance is penalized with the diagonal length of the map, l. The distance from r to a contour cell c_i of RI is defined as:

$$d(r,c_i) = g(r,c) + \rho \qquad c_i \in RI \tag{2}$$

The penalizing value ρ will give priority to the contour cells that do not have an obstacle blocking the direct line between the robot and them. Thus, ρ is 0 if there is no obstacle between r and c, otherwise, it is equal to l.

Finally, the distance between r and R is calculated as:

$$d(r,R) = \min_c\{d(r,c)\} \qquad c \equiv c_a \quad or \quad c \equiv c_i \tag{3}$$

Having defined the distance from a robot to a region, it is possible to determine the closest region to a robot. In some cases, two or more robots are closer to the same region than to other regions. If there were no control on the assignment of regions, all these robots could end up exploring the same region. Therefore, it is necessary to design an optimal solution to assign robots to regions fairly. In this proposal, the assignment between robots and regions is formulated as a Linear Programming (LP) problem. An LP solver [11] is applied to obtain an optimal solution.

If there are N robots, the optimization objective consists of finding out an N-tuple of robot and region pairs $((r_1,R'_1),(r_2,R'_2),\ldots(r_n,R'_n))$ such that the total sum of robot to region distances is minimized. R'_i represents the region that is assigned to r_i.

Let $a_{ij} = 1$ denote that region R_j is assigned to r_i, and $a_{ij} = 0$ that R_j is not assigned to r_i ($i,j = 1,\ldots,N$). The problem is defined as finding a set of a_j ($i,j = 1,\ldots,N$) subject to:

$$\sum_{i=1}^{N} a_{ij} = 1, \qquad \sum_{j=1}^{N} a_{ij} = 1 \qquad i, j = 1,.., N \tag{4}$$

that minimize the sum of distances between robots and regions:

$$\sum_{j=1}^{N} \sum_{i=1}^{N} a_{ij} d(r_i, R_j) \tag{5}$$

This optimal assignment ensures that the total sum of path lengths of robots moving to their assigned regions is minimized. In the case of Fig.3, r_1, r_2 and r_3 are assigned to R_1, R_2 and R_3. The assignment of regions to robots is adjusted during the exploration so as to minimize the costs of driving the robots in order to explore their corresponding regions. This optimization algorithm is iteratively applied since the relative positions between robots and regions keep changing as new areas of the workspace are being discovered.

C. Robot-Target Pair Assignment

Having decided the assignment of a robot to every region, this section describes to which frontier cell each robot should travel. If the robot's region is accessible for it, it will go to a frontier cell that is beside the closest contour cell of its own region. If a robot's region is not accessible for it, since it will have to travel through the regions of other robots, it is necessary to decide which frontier the robot should visit first.

To reduce the large number of frontiers in the exploration map, a set of M frontiers is selected, FM, such that each frontier in FM stays away from each other at least the sensor range. Each robot should then choose a frontier from FM different from the frontiers chosen by the other robots.

Let R_{r_i} be the optimal region assigned to robot r_i according to the scheme described in Section IV.B, and let cc_{r_i} be the contour cell of R_{r_i} closest to r_i. In the case of Fig. 3, c_1 is cc_{r_1} of R_1 for r_1, c_3 is cc_{r_3} of R_3 for r_3.

The goal of the current assignment is to evaluate all the candidate frontiers from FM for every robot r_i and find the best target for it. Denote each candidate frontier as f_j ($j = 1, 2,\ldots,M$), and the target frontier for r_i as T_i.

Since the robots whose regions are inaccessible for them need to reach their regions as soon as possible, they have priority to choose their targets first. For those whose regions are inaccessible, equation (6) evaluates the distance between a robot ri and each frontier of FM, $d(r_i, f_j)$. Its value is determined if f_j is accessible for r_i from the robot's position. Otherwise, it is infinite.

$$d(r_i,f_j) = \begin{cases} \delta(r_i,f_j) + \rho(f_j,cc_{r_i}) & if \;\; f_j \;\; is \;\; accessible \;\; for \;\; r_i \\ \infty & otherwise \end{cases} \tag{6}$$

where $\rho(f_j, cc_{r_i}) = g(f_j, cc_{r_i}) + \rho_1(f_j, cc_{r_i}) + \rho_2(f_j)$

In (6), δ denotes the real path cost. Since it is possible that the space between a candidate frontier f_j and a robot's closest contour cell cc_{r_i} is unexplored (in Fig.3, T_1 is a frontier cell and between T_1 and c_1 there is unexplored space), the geometric distance g is used. Functions ρ_1 and ρ_2 add penalty values to the distance $d(r_i, f_j)$ when f_j is not favorable for r_i.

If there is an obstacle along the line from f_j to cc_{r_i}, $\rho_1 = l$, otherwise $\rho_1 = 0$. If f_j has been chosen as a target by another robot, $\rho_2 = v$, otherwise $\rho_2 = 0$.

Constant v is a positive value. Adding it will force robots not to choose the same frontiers as the ones which have already been chosen by other robots. Constant l is the same diagonal length mentioned in Section IV.B.

The penalty values distinguish the frontiers from each other. If there are two frontiers (f_1, f_2) which are close to each other, and there are obstacles lying on the lines from a robot's closest contour cell cc_{r_i} to f_1 and f_2, they will both get a penalty value l, such that the one with the smallest distance will win. If there is another f_3 which does not have an obstacle lying on the line from it to cc_{r_i}, it will win both f_1 and f_2.

The above evaluation process starts with the first robot r_i (robots are ordered from r_1, to r_N) whose own region is inaccessible. For each robot, the frontier whose distance to the robot is the smallest will be chosen as the target T_i for it. For instance, in Fig.3, T_1 and T_3 are assigned to r_1 and r_3 respectively.

$$T_i = \arg\min_j (d(r_i, f_j)) \qquad (7)$$

In the next step, equations (6)-(7) are used again for finding the target frontier for robots whose regions are accessible. There is no much difference between the selection process for robots whose regions are accessible and inaccessible, since $g(f_j, cc_{r_i})$ will make a robot tend to choose the frontier which belongs to its own region and close to the closest contour cells of its own region. Therefore, any robot r_i whose own region is accessible will probably go to the frontier beside cc_{r_i}. Only in the case that there is a frontier lying on the way to cc_{r_i}, the robot will visit it.

D. Exploration Algorithm

The proposed exploration approach is described below:

```
1:  Initialize exploration map, real path map, and K-means label map.
2:  **While** there are unexplored cells **do**
3:  **If** iteration step is lower than 2, **then** initialize centroids of K-
    Means randomly from the workspace
4:  Search for frontiers beside unknown regions, extract a set of $FM$
    from the frontiers.
5:  Calculate the real path distance from robots to all free cells and
    store them in robots' real path map
6:  Call K-Means to partition the unknown cells and frontier cells
    and label these cells by the region identifiers in K-means label
    map
7:  Calculate one distance between robots and regions
8:  Call LP-solver to assign robots to regions
9:  For robots whose regions are inaccessible, calculate the distance
    from each robot to each frontier, find the smallest distance and
    assign the correspondent frontier to the robot
10: For robots whose regions are accessible, calculate the distance
    from each robot to each frontier, find the lowest distance and
    assign the correspondent frontier to the robot
11: Robots move to their targets until one robot arrives at its target
12: Robots sensor the environment and update the exploration map
13: **end while**
```

V. Experiments

The multi-robot exploration algorithm proposed in this paper and three alternative approaches representative of the state-of-the-art (Yamauchi [3], Burgard et al. [4], [5], Zlot et al. [6]) have been extensively tested in simulation. Sections V.A to V.C describe the experimental setup, the concepts of regional waiting time and waiting time variance, and the experimental results.

A. Experimental Setup

The types of workspace on which the experiments have been carried out are illustrated in Fig.4. The four algorithms are run on all three maps with two starting schemes for the robots: (a) all robots start from the same location, (b) robots start from random locations (see Fig. 5(left)(middle)). Two to eight robots have been tested in order to find out the influence of the number of robots on the performance of the algorithms when the workspace is divided into different number of regions. For every combination of robots, maps and starting schemes, 15 set-ups of runs have been performed. When robots start together, they do it from one of the 15 locations shown in Fig. 5(right). When robots start with the separate scheme, the robots' initial positions are randomly chosen from those 15 locations.

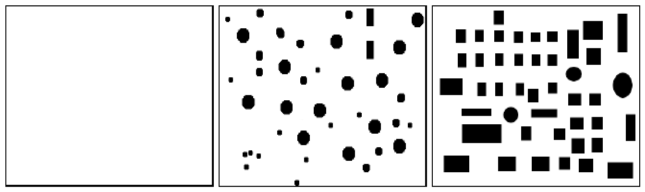
Fig. 4. Three types of workspace (maps): (left) blank map (middle) unstructured map (right) cityblock map.

An experimental configuration includes the combination of a number of robots between 2 and 8, a map out of three, a starting scheme out of two and a starting location out of 15. Hence, the four tested exploration algorithms have been compared over 630 different configurations (3 maps x 2 starting schemes x 7 applied numbers of robots x 15 runs).

The objective of the experiments is to find out how the four exploration approaches behave with respect to the criteria of waiting time variance and exploration percentage variance. The two criteria are defined in the next subsection.

B. Waiting Time Variance and Exploration Percentage Variance

When a robot arrives at a cell at step t and explores its surrounding area with its sensor, suppose cell p is explored, no matter if it is found free or occupied. The *waiting time stamp* (w) of p is set to t. At the end of the exploration, each cell is marked with a definite *w*. At each step *t*, some cells have already been explored (known cells), whereas the other cells have not (unknown cells). *w* is recorded for every known cell. For every unknown cell, the *estimated waiting time stamp* (w') is calculated as ($w' = e + t$), where *e* is the number of estimated steps from a closest robot to the unknown cell (the distance divided by the cell size).

Let *R* be a region of the workspace that contains *n* known cells and *m* unknown cells. The regional waiting time w_R of that region at time *t* is defined as:

$$w_R = (\sum_{i=1}^{n} w_i + \sum_{j=1}^{m} w'_j)/(m+n) \quad (8)$$

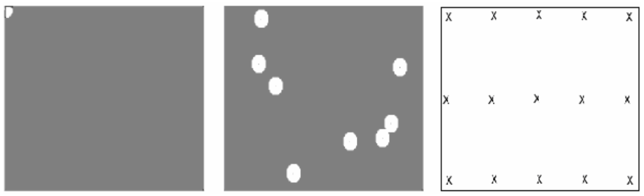
Fig. 5. Starting schemes: (left) robots start together, (middle) robots start from random separate places. Gray represents cells that are unknown for robots. White represents cells that have been explored by the robots' sensors and have been found to belong to the free space. (right) 15 set-ups of the robots' starting positions.

Without loss of generality, the maps are divided into 2, 4, 8, 16, 32, 64 equal-sized regions and the program calculates w_R for each region. The variance of regional waiting time of *N* regions is simply defined as:

$$WTV = \sigma^2(W_{R_i}), \quad i=1,...,N \quad (9)$$

The lower the WTV is, the more balanced the waiting time of regions is. When the exploration process terminates, every cell of the workspace is known and has a waiting time *w* value. The final average waiting time for a region *R* with *n* cells becomes: $awt_R = (1/n)\sum_{i=1}^{n} w_i$

The final WTV becomes the variance of average waiting times of all regions:

$$WTV_{final} = \sigma^2(awt_{R_i}), \; i=1,...,N \quad (10)$$

The exploration percentage of a region (*EP*) at each time step is simply how much percentage has been explored in that region. The variance of *EP* of all regions (*EPV*) gives an explicit measure of the concurrency in exploring the different parts of the workspace.

C. *Experimental Results*

Fig. 6 shows the mean WTV of eight regions for 15 runs on the unstructured map and the together scheme considering the four tested exploration approaches. YMC, Burgard, Zlot and KME represent Yamauchi [3], Burgard et al. [4], [5], Zlot et al. [6] and the proposed algorithm respectively. The four approaches have been compared under the same configurations. The WTV curves show that the more robots, the lower the WTVs of all approaches. When more than two robots are utilized, KME has the lowest WTV during the exploration, meaning that it explores the different regions with more similar regional waiting times. In addition, WTV_{final} of regions (the value where each curve terminates) with KME is also the smallest except for the two-robot case, which means that the *awt* of different regions are the most similar with KME.

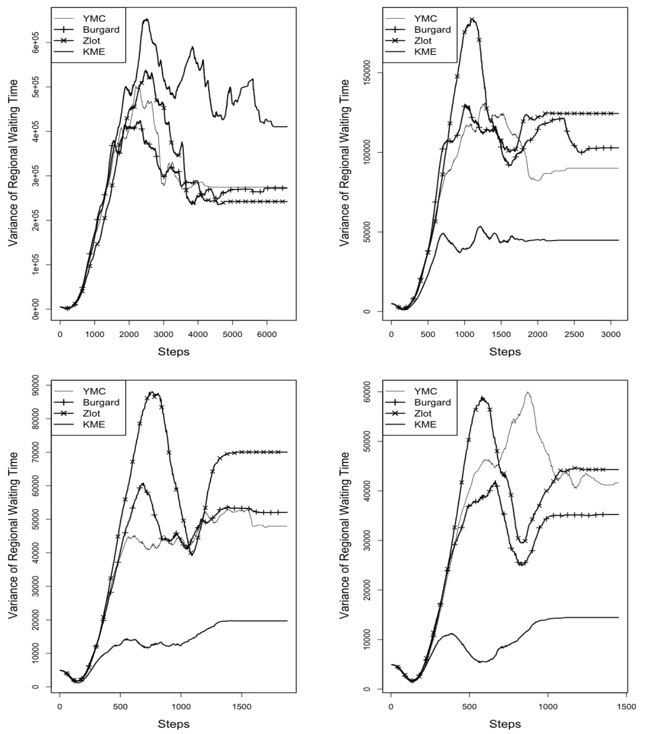

Fig. 6. Curves (mean of 15 runs) of WTV of eight regions and exploration scheme "unstructured map / robots start together". Subfigures from left to right respectively show the curves for 2, 4, 6, and 8 robots.

The figures for the same experiments with the workspaces being divided into two, four, sixteen, thirty-two and sixty-four regions are not shown in the paper for space limitations and can be found in the following link (http://deim.urv.cat/~rivi/KMEfigures.html)

From the comparison of WTVs for different divisions of regions, it can be noticed that with the same number of robots (2, 4, 6 and 8 robots), as long as the number of regions is not much bigger than the number of robots, KME always leads to the lowest waiting time variance of all the regions. Hence, the waiting time of all regions is more equalized than with the other three approaches. The same trend can be found in the six combinations of maps and starting schemes.

Fig. 7 shows the results for the unstructured map workspace, and 2, 4, 6, and 8 robots starting from separate locations. The WTV curves of all techniques with the separate scheme are lower than the ones with the together scheme. This is reasonable as when robots are initially separate, they naturally disperse. Thus, the regional waiting time among regions is more equalized. With the separate scheme, KME is again superior to the other methods.

In the figures discussed above, each curve represents the average of the 15 runs. In order to show that a curve obtained by averaging the WTV curves of 15 runs is general enough to observe the general trend, 100 experiments have been carried out under the same conditions. In each of the 100 runs, the robots start from a random location. The mean of WTV curves for 1, 5, 10, 15, 20, 30, 50 and 100 runs with the workspace divided into 16 regions have been drawn (see the aforementioned link). They show the same trend as the above figures. KME still has the lowest WTV in those experiments. The curves become smoother when the number of runs increases. The only exception is the Zlot's method, which grows beyond the Yamauchi's method when more than 30 runs are averaged.

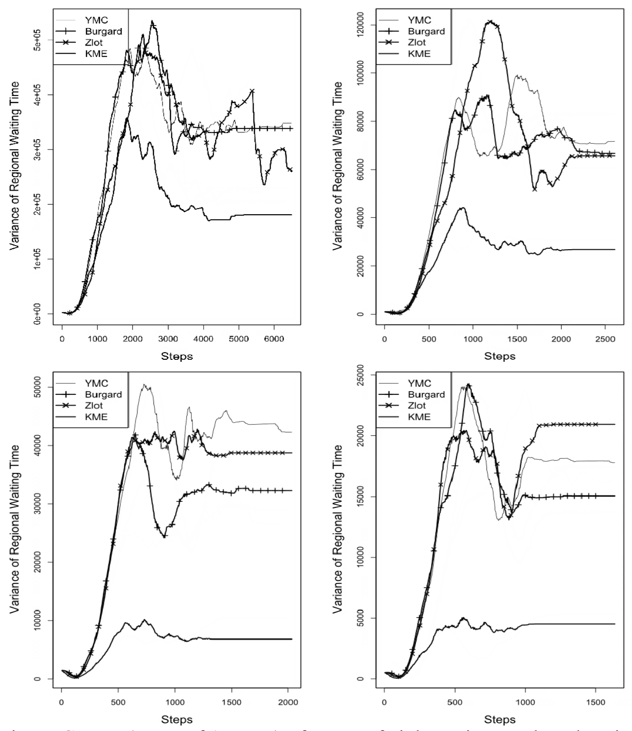

Fig. 7. Curves (mean of 15 runs) of WTV of eight regions and exploration scheme "unstructured map / robots start separately". Subfigures from left to right respectively show the curves for 2, 4, 6, and 8 robots.

Fig. 8 shows the experiments with respect to EPV with 2, 4, 6, and 8 robots exploring the unstructured map with the robots starting from different locations. The four approaches have been compared with respect to the exploration percentage of eight regions under the same conditions (same map, starting scheme and starting locations). Each subfigure illustrates the mean of EPV of 15 runs during the exploration. Notice that with 2, 4, 6, and 8 robots, the eight regions are explored at a similar speed with KME, whereas even with more than 2 robots, the EPV curves of the other three methods show a big difference of exploration percentage between two regions (see more results in http://deim.urv.cat/~rivi/KMEfigures.html).

Notice that no matter the number of regions, KME is the approach that explores the different regions with the most equalized speed. When the workspace is divided into 2^n regions, if the number of robots is lower than n, KME gradually behaves similarly to the other methods with respect to EPV. Hence, when the number of robots is bigger than n and the number of regions is not too high compared to the number of robots, KME still performs better than the other methods.

On the other hand, Zlot's algorithm behaves like [3]-[5] with respect to the WTV and EPV. It is clear to see from the descriptions of the methods [3]-[5] that the robots do not disperse or disperse in just a local level. The results presented here show that the Zlot's algorithm does not force robots to disperse into the whole workspace more than the other algorithms [3]-[5]. Although it applies an optimization method for task assignment, its solution is not superior to [3]-[5] with respect to the waiting time variance.

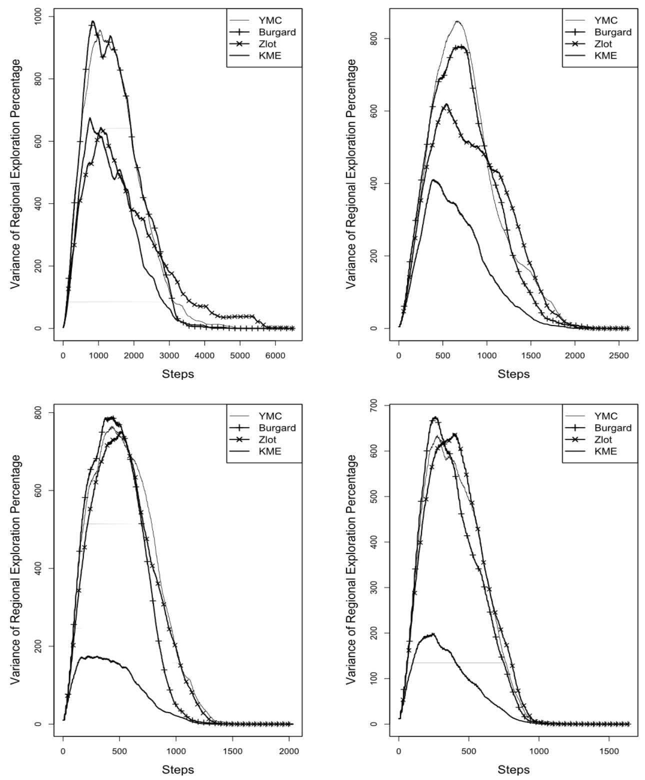

Fig. 8: Curves (mean of 15 runs) of EPV for eight regions and exploration scheme "unstructured map / robots start separately". Subfigures from left to right respectively show the curves for 2, 4, 6, and 8 robots.

VI. Conclusions

A new algorithm (KME) for multi-robot exploration has been presented in this paper. The proposed algorithm leads to the lowest variance of average waiting time of regions when compared to the state of the art approaches tested in this work, that is, different regions are explored with the most similar average waiting time, this being an important benefit when an application requires a balanced exploration of different regions of the workspace with a number of robots inferior to that number of regions. This behavior is desirable for many exploration applications, such as search and rescue.

References

[1] M. Mataric, "Minimizing complexity in controlling a mobile robot population", *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, vol. 1, 1992, pp. 830-835.

[2] A. Solanas, M. A. Garcia, "Coordinated multi-robot exploration through unsupervised clustering of unknown space", *Proc. Int. Conf. on Intelligent Robots and Systems*, vol.1, 2004, pp. 717-721.

[3] B. Yamauchi, "Frontier-based exploration using multiple robots", *Proc. of Int. Conf. on Autonomous Agents*, 1998, pp. 47-53.

[4] W. Burgard et al., "Collaborative multi-robot exploration", *Proc. Int. Conf. on Robotics and Automat.*, vol.1, 2000, pp. 476-481.

[5] W. Burgard, M. Moors, C. Stachniss, F. Schneider, "Coordinated multi-robot exploration," *IEEE Transactions on Robotics*, vol. 21, no. 3, 2005, pp. 376-386.

[6] R. M. Zlot, A. T. Stentz, M. B. Dias, S. Thayer, "Multi-robot exploration controlled by a market economy", *Proc. Int. Conf. on Robotics and Automation*, vol. 3, 2002, pp. 3016-3023.

[7] D. Fox et al., "Distributed multirobot exploration and mapping," *Proceedings of the IEEE*, vol. 94, 2006, pp. 1325-1339.

[8] R. Simmons et al., "Coordination for multi-robot exploration and mapping", *Proc. Conf. on Artific. Intel.*, 2000, pp. 852-858

[9] D. P. Bertsekas, "The auction algorithm for assignment and other network flow problems: a tutorial," *INTERFACES*, vol. 20, no. 4, 1990, pp. 133-149.

[10] N. Ye, X. Li, T. Farley, X. Xu, "Job scheduling methods for reducing waiting time variance," *Computers and Operations Research*, vol. 34, no. 10, 2007, pp. 3069-3083.

[11] Lp-solve, Available: http://lpsolve.sourceforge.net/5.5/

[12] L. Wu, M.A. Garcia, D. Puig, and A. Sole, "Voronoi-based space partitioning for coordinated multi-robot exploration," *Journal of Physical Agents*, vol. 1, no1, July 2007, pp. 37-44.