Multi-Robot Simultaneous Coverage and Mapping of Complex Scene - Comparison of Different Strategies

Robotics Track

Laetitia Matignon* †
* Univ Lyon, CNRS
Universite Lyon 1, LIRIS, UMR5205, F-69622, France
laetitia.matignon@univ-lyon1.fr

Olivier Simonin† ‡
† CITI Lab., Inria Chroma team, F-69680, France
‡ INSA Lyon, Universite de Lyon, F-69100, France
olivier.simonin@inria.fr

ABSTRACT

This paper addresses the problem of optimizing the observation of a human scene using several mobile robots. Mobile robots have to cooperate to find a position around the scene maximizing its coverage. The scene coverage is defined as the observation of the human pose skeleton. It is assumed that the robots can communicate but have no map of the environment. Thus the robots have to simultaneously cover and map the scene and the environment. We consider an incremental approach to master state-space complexity. Robots build an hybrid metric-topological map while evaluating the observation of the human pose skeleton. To this end we propose and evaluate different online optimization strategies exploiting local versus global information. We discuss the difference of the performance and cost. Experiments are performed both in simulation and with real robots.

KEYWORDS

multi-robot systems; mapping and exploration; networked robot/sensor systems

ACM Reference Format:
Laetitia Matignon* † and Olivier Simonin† ‡. 2018. Multi-Robot Simultaneous Coverage and Mapping of Complex Scene - Comparison of Different Strategies. In *Proc. of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018), Stockholm, Sweden, July 10–15, 2018,* IFAAMAS, 9 pages.

1 INTRODUCTION

Research in complex scene observation by mobile robots has recently gained significant attention and practical interest. A key direction of this research aims at coordinating robots to explore the environment and at optimizing their positioning so as to maximize the quality of the scene observation. This is especially challenging when human activities must be observed, or monitored, in complex structured environments. This concerns for instance cobotics and rescue tasks.

In this paper we focus on observing a person carrying out an activity in a specific area. Several robots have to deploy themselves around the person to fully observe its pose (*i.e.* identify its skeleton pose). As the environment is unknown to the robots and cluttered, robots have to explore it in order to find the best joint observation. We assume that robots are homogeneous, can communicate and know the relative position of the scene to observe.

Several strategies have been proposed to coordinate robots in tracking a set of targets. They range from very simple local rules to heuristic-global approaches, see review [9]. However these solutions generally consider that the environment is free of obstacles, or they are too few to obstruct the observation.

In this paper we address multi-robot scene observation in unknown cluttered environment. This raises new questions: how to map the environment while searching for an optimal positioning ? how to limit the state space to map and to explore ? how to represent the quality of observation from each location visited by the robots ?

To this end we build an approach which extends the coverage (observation) of the targets by considering to simultaneously map the environment. Then we tackle another issue which is the trade-off between exploitation and exploration, that is moving to optimize the observation versus exploring the environment to find new interesting observation positions.

In response to these questions we propose an incremental mapping that refines the representation and the information in areas where the quality of observation is promising. We also introduce a circular topology adapted to the continuous observation of the scene while robots move around. Then we propose different algorithms relying on local versus global information, leading to solutions with different computational costs. We conduct a set of experiments in simulation and with real robots allowing to evaluate the different approaches.

The paper is organized as follows. In Section 2 we discuss related works on the tracking of targets with mobile robots. Section 3 formalizes the problem and defines the coverage and mapping tasks. Then we propose in 4 different algorithms relying respectively on random decisions, meta-heuristics and brute-force search. Section 5 presents a simulator and some experiments comparing the performances of the different approaches. Finally, we conclude and draw some perspectives to this work in section 6.

2 RELATED WORK

The last decade have seen a growth of the research on network of fixed cameras to detect, track and recognize objects or persons [1, 16]. However, a set of static cameras cannot deal with non-covered zones or occlusions. That's why recent

Proc. of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018), M. Dastani, G. Sukthankar, E. André, S. Koenig (eds.), July 10–15, 2018, Stockholm, Sweden.

works in the field of **active perception** are interested in using mobile cameras that can move to adequate places to cover blind spots or to react to changing conditions as lighting or dynamic obstacles.

Cooperative active perception considers multiple decision makers that cooperate and merge information from their sensors. In the URUS[1] project [15], the objective is to assist and guide people in urban settings. Active cooperative perception has been here considered only between one mobile robot and a set of fixed surveillance cameras. In Giusti et al. work [4], distributed visual recognition of hand gestures is carried out with a group of mobile robots. They use a distributed consensus protocol to decide, between individual classifications made by each robot, the issued gesture. In this work, the navigation and coordination of the robots is very simple and is not used to improve the recognition. Indeed, although the robots are mobile, once they are positioned uniformly in a semi-circular arc centered on the target scene, they maintain this formation. Moreover, the environment is assumed to be without obstacles, which facilitates the navigation and the observation.

Exploration and mapping of cluttered environments with autonomous mobile robots have been intensively studied since two decades. The objective is to explore as quickly as possible all the area of the environment in order to build a complete map. Using a fleet of robots instead of a single one allows to divide the task, leading to a gain of time, robustness and accuracy, see e.g. [3, 8, 12, 14]. However, these works do not consider that the mapping has to be conduced in order to help the positioning and coordination of robots so as to optimize the observation of a scene, which is our problem.

Approaches using the navigation and coordination of multiple robots to perform cooperative active perception can be found in works about the observation of moving targets with moving sensors. Most of these approaches are classified and discussed in a recent review [9]. Among these, **the CMOMMT (Cooperative Multi-robot Observation of Multiple Moving Targets)** framework, introduced by L. Parker [13], aims to dynamically position robots to maximize the number of targets under observation and the duration of observation of each target. An on-line and distributed heuristic approach is proposed to solve the CMOMMT problem using weighted local force vector control. However this approach assumes uncluttered environments with either no or simple obstacles. Moreover it is supposed that the navigation behaviour is only influenced by the obstacles, and not by the perception of the targets. This means that the robots observe the targets even if there are obstacles and they do not observe the targets only when they are too far from them. So this approach is not adapted to our context where robots have to evolve in an unknown cluttered environment composed of obstacles affecting the navigation and the observation of the scene. However we build our approach as an extension of the CMOMMT model, and we use its formalism to define our problem in the following section.

[1]Ubiquitous Networking Robotics in Urban Settings.

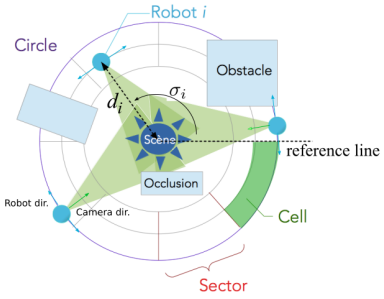


Figure 1: Joint observation (coverage) of a scene (human activity) with a fleet of $m = 3$ robots. The navigation model is based on circles with a spatial discretization in cells ($C = 2$ circles and $D = 8$ sectors).

3 PROBLEM DEFINITION

In this section we formally define the generic task of the multi-robot observation of a human activity in an unknown environment. We consider in this paper a person carrying out an activity in a quasi-static location. The observation is defined as identifying the human body pose, that can be characterized by a set of skeleton joints. We give general definitions before refining the mapping and the coverage tasks by considering a specific circular navigation topology, more adapted to the observation around a scene.

3.1 Coverage and Mapping

We formalize the complex scene observation problem as an extension of the CMOMMT[2] framework [13]. This framework aims to dynamically position robots to maximize the **coverage** of mobile targets, i.e. the number of targets under observation and the duration of observation of each target.

Definition 3.1. The **CMOMMT** model [13] is defined as a tuple $< S, \mathcal{V}, K >$ where:

- S a two-dimensional, bounded spatial region;
- $K(t)$ a set of n targets, where $\kappa^t_{j,j=1,\ldots,n}$ is a target, that is located within region S at time t; in our case, the human scene to observe is composed of a set of n skeleton joints;
- $\mathcal{V}$ a team of m mobile robots, where $\nu_{i,i=1,\ldots,m}$ is a robot with observation sensors that are potentially noisy and of limited range.

Observing human(s) activity in an unknown and structured environment requires the robots to build a map. The map will allow the robots to locate themselves, to share information, to plan paths among obstacles, and to learn worst versus best observation locations. Thus **we complete the CMOMMT framework and its coverage task with a simultaneous mapping task**.

To master the complexity of the state space, we propose **a discrete representation of the environment and the robots' positions**. We define a grid of $D \times C$ contiguous

[2]Cooperative Multi-robot Observation of Multiple Moving Targets

cells where C is the number of concentric circles around the scene, divided in D sectors. Then the robots move in discrete-circular topology by following the C circles or changing of circle, as it is illustrated in Fig. 1.

Definition 3.2. The **position** of a robot ν_i at time t is defined by $x_i^t = (d_i(t), \sigma_i(t))$ with d_i the distance of the robot ν_i to the scene, and σ_i the angle between a reference line and the line connecting the scene to the robot (cf. Fig. 1).

Definition 3.3. At any position of a robot ν_i at time t is associated a unique **cell** $c_i^t = \langle [d_a, d_b]; [\sigma_a, \sigma_b] \rangle$ such that $d_i(t) \in [d_a, d_b[$ and $\sigma_i(t) \in [\sigma_a, \sigma_b[$. The **position of a cell** $c = \langle [d_a, d_b]; [\sigma_a, \sigma_b] \rangle$ is defined as $(d_a, \frac{\sigma_a + \sigma_b}{2})$.

This concentric topology allows the robots to maintain their direction of observation towards the scene while they move along circles. This also reduces dramatically the number of position of observation. Some of these positions can be unreachable when occupied by obstacles or other robots.

3.2 Observation data and Actions

Skeleton observation. In the considered observation problem, a target is a skeleton joint. A joint is observed when a robot is able to identify the joint with its sensor.

Definition 3.4. The **observation vector** $o_i(x_i^t)$ of a robot ν_i at time t is defined as a binary vector of size n such that:

$$o_i(x_i^t) = [o_{ij}(x_i^t)]_{j=1..n} = [o_{i1}(x_i^t), ..., o_{in}(x_i^t)] \quad (1)$$

where:

$$o_{ij}(x_i^t) = \begin{cases} 1 & \text{if robot } \nu_i \text{ is observing target } \kappa_j^t \\ & \text{from its position } x_i^t \\ 0 & \text{otherwise.} \end{cases}$$

Definition 3.5. The **individual observation quality** $q_i(x_i^t)$ made by a robot ν_i at time t is defined as:

$$q_i(x_i^t) = \frac{1}{n} \sum_{j=1}^{n} o_{ij}(x_i^t). \quad (2)$$

The quality $q_i(x_i^t)$ is the percentage of skeleton joints accurately tracked by the robot ν_i at t.

Definition 3.6. The **observation matrix** O of the team $\mathcal{V}$ at time t [13] is defined as :

$$O(x_1^t, ..., x_m^t) = [o_i(x_i^t)]_{i=1..m} = [o_{ij}(x_i^t)]_{i=1..m;j=1..n}$$

Definition 3.7. The **joint observation quality** Q made by the team $\mathcal{V}$ at time t is defined as:

$$Q(x_1^t, ..., x_m^t) = \frac{1}{n} \sum_{j=1}^{n} g_j(O(x_1^t, ..., x_m^t)) \quad (3)$$

where:

$$g_j(O(x_1^t, ..., x_m^t)) = \begin{cases} 1 & \text{if } \exists i \in \mathcal{V} \text{ such that } o_{ij}(x_i^t) = 1 \\ 0 & \text{otherwise.} \end{cases}$$

To quantify the individual contribution of each robot to the joint observation, we introduce the notion of **marginal contribution**. This refers to the marginal contribution of a player to a coalition in the Shapley value [6].

Definition 3.8. The **marginal contribution** w_i of a robot ν_i in the joint observation of the team, at time t, is defined as:

$$w_i(x_1^t, ..., x_m^t) = \frac{1}{n} \sum_{j=1}^{n} o_{ij}(x_i^t) \wedge (o_{ij}(x_i^t) \oplus g_j(O(x_{k,k\neq i}^t))) \quad (4)$$

with $\oplus$ the exclusive disjunction.

The marginal contribution corresponds to the part of the observation that robot ν_i is the only one to see. It depends on the positions of all the robots of the team.

Coverage criterion example. The objective is to maximize the joint observation quality Q, i.e. the number of targets observed by the team. Notice that maximizing Q is not decomposable into maximizing q_i for each robot, which could lead to redundant information. For instance, consider the following example with $m = 3$ robots and $n = 7$ targets. Observation vectors, individual qualities and marginal contributions of each robot at time t are:

$$o_1^t = [0, 0, 0, 1, 1, 0, 0],\ q_1^t = 0.29,\ w_1^t = 0.29$$
$$o_2^t = [1, 1, 0, 0, 0, 1, 1],\ q_2^t = 0.57,\ w_2^t = 0$$
$$o_3^t = [1, 1, 1, 0, 0, 1, 1],\ q_3^t = 0.71,\ w_3^t = 0.14$$

Even if Robot ν_2 has a high individual observation quality, its contribution is low because it observes the same targets as another robot. Conversely, ν_1 has a low individual quality, but it is the only one to observe some targets, so its contribution is the highest. This illustrates that the objective of maximizing Q requires to find the most complementary information.

Map data. In each **cell** c of the map we compute the following data :

- an obstacle occupancy probability (computed from the robots' SLAM[3] function);
- a number of visits (considering all the robots);
- the set of observation vectors done so far from that cell. The size of this set is equal to the number of visits.

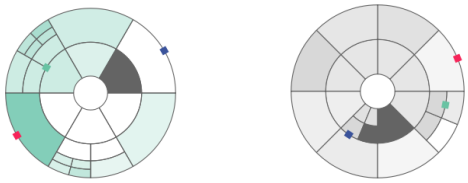
From this, we can compute[4] the **mean and rounded mean observation vector** $\tilde{o}(c)$.

Definition 3.9. The **mean observation vector** $\tilde{o}^t(c)$ of the cell c at time t is the sum of the observation vectors done so far, divided by the number of visits of that cell until t.

Definition 3.10. The **rounded mean observation vector** of the cell c at time t is the mean observation vector where each elements of the vector is rounded to 1 or 0 according to its comparison to 0.5.

Incremental division of cells. To master the time to explore the environment, we propose an **incremental division of cells** based on a quadtree structure. The idea is to have at the beginning a coarse representation of the environment with few initial cells. A leaf cell can be split into four

[3]Simultaneous Localization and Mapping
[4]Incremental averaging is used.



(a) Coverage quadtree data. (b) Obstacle quadtree data.

Figure 2: Two quadtree maps constructed by 3 robots (colored squares) on different environments. Map data are: (a) cell qualities in shades of green (the greener the cell, the better is the local observation; white is for cells where the scene is not visible, dark for obstacle cells) (b): obstacle probability in shades of grey.

sub-cells (sub-cells become the new leaf cells), and recursively, sub-cells can be split until the max depth of the quadtree is reached. This is illustrated in Fig. 2. The objective is to **refine the discretization only in interesting areas of the environment**. Thus the number of cells to explore is reduced while refining the joint observation quality over time as the robots split interesting cells.

Robots actions. Robots can execute two kind of **actions**:

- split its current cell ; then the robot goes randomly to one of the four children cells;
- move to a cell: move to an adjacent cell or compute a path to a specific cell. Robots are moving on cells which are leaf nodes of the quadtree.

Cell's data are **updated** following probabilistic quad-tree principle [11]. When a cell is split, all the data of the children cells are initialized with the data of their parent node, except the obstacle occupancy probability that is computed from the occupancy grid map.

Exploration-Exploitation compromise. The problem is defined as two concurrent tasks, which are mapping the environment and covering the scene. The objective is to reach as soon as possible a robots' joint position that maximises the coverage criterion. **The exploitation** consists in using the information gathered by robots in each cell, i.e. in the map, to find this joint-position.

In all cases, we need to **explore** at the beginning of the task to acquire and gather cells data. Then, **a compromise between exploration and exploitation** is required. The compromise is between the quality of the found solution (the more we explore cells and gather data, the more cells data will be pertinent), and the time to find a good solution (how long do we need to explore to gather pertinent enough data). Especially if the scene is dynamic and the targets are moving, robots do not have a lot of time to find a solution.

4 ALGORITHMS

In this section we present different strategies for the simultaneous exploration of the environment and coverage of the scene. In each strategy, or algorithm, robots/agents are homogeneous and supposed at least to communicate their location and split actions. These algorithms do not exploit the same data and do not compute/store the same information. We discuss in the end of the section their differences in term of computational and memory cost, before comparing their performances in the next section.

4.1 Heuristics algorithms

Consider the state space that m robots have to explore in order to find the best position. This set is bounded[5] by $(C \times D)^m$ if robots are heterogeneous. If robots are identical[6], the size of the state space goes down to $\binom{C \times D}{m}$. Given the obstacles, it does not reduce to $\binom{D}{m}$ as it is not sufficient for the robots to explore the cells situated the closest to the scene as they are not guaranteed to be reachable. To handle this complexity we **combine the incremental mapping with heuristics algorithms to guide the exploration of the state space**.

In general, exploration consists in moving to unknown cells to gain new information and to avoid possibly the team remaining in a local optimum; while exploitation consists in moving to known cells to optimize the joint observation. Here **exploration** consists for a robot in moving to one of its unvisited adjacent cells. If there are no unvisited adjacent cells, the robot tries to move to adjacent cells by choosing first the less visited ones. **Exploitation** consists for a robot in moving to its best adjacent cell, or split its current cell if it is already the best one or if moving is not possible[7]. The best adjacent cell is defined for a robot as the cell that, among the current cell of the robot and its adjacent cells, maximizes the joint quality given the current cells of the other robots. The rounded mean observation is used to compute the joint quality and especially, to infer the observations made from the adjacent cells.

We propose different algorithms based on two major steps:

- The first one is the **selection of which robots must move at each decision step**. The proposed heuristic is to move only the robots with the **lowest marginal contributions**. Keeping static the robots with high contributions still offers several advantages: minimizing the decay in the current joint quality, maintaining the group configuration stable, and detecting potential changes in the scene (activity change).
- The second one is to determine **which action must be done by these selected robots**. To this end we study different strategies:

[5]It can be reduced if some cells are inaccessible because of obstacles and if only one robot per cell is allowed.
[6]Two robots can be arbitrarily swapped without changing the joint observation of the scene.
[7]A move is not possible if the target cell is an obstacle or is already occupied by another robot.

- `RandomAction` chooses to split the current cell of the robot with a probability of 0.5, and otherwise to move to one of the adjacent cells of the robot chosen randomly;
- `SimulatedAnnealing` (SA) [10] chooses an action of exploration according to a probability τ, and an action of exploitation otherwise. This temperature parameter is gradually reduced by a decreasing rate $\alpha \in [0; 1]$ at each decision step t according to: $\tau_t = \tau_{t-1} \times (1 - \alpha)$;
- `TabuSearch` (TS) [5] always performs an action of exploitation, but forbids the visit of the cells that are in the tabu list. The tabu list keeps tracks of a short-term set of the last visited cells.

Two main algorithms are obtained combining the options at each step. Algorithm 1 chooses randomly which robots to move at each step, and their actions are also selected randomly. Algorithm 2 gathered various algorithms where only the robots with the lowest marginal contributions move at each step. In Algorithm 2, each moving robot can choose its action according to `RandomAction` (it gives the `RandomWithContribution` algorithm) or meta-heuristics. In particular, using meta-heuristics as `TS` or `SA` leads to a behaviour where the cells that are potentially interesting for the observation of the scene are divided and explored more accurately.

Algorithm 1: One step of `Random` search

Data: The set of m robots,
the number of moving robots $\delta \in [1..m]$
for $i \in [1, .., m]$ **do**
  Update o_i
`// 1- Choose which robots to move`
$movingRobots \leftarrow$ choose randomly δ robots among m
`// 2- Choose an action`
for $i \in [1, .., \delta]$ **do**
  $c_i \leftarrow$ current cell of the robot $movingRobots[i]$
  $movingRobots[i]$ executes `RandomAction`(c_i)

4.2 Exhaustive search (brute-force)

To compare our heuristic algorithms with a brute-force approach, we propose an exhaustive combination algorithm. It aims to find the best joint position around the scene by computing the qualities of all the possible joint positions, given the cells already visited by the robots.

4.2.1 Combination definitions. A **combination from the set** m is defined as a set of m different leaf cells and noted λ. It represents a joint position for the m robots. The **quality of a combination** at step t is defined as the joint quality computed with rounded mean observations of the cells.

The **cost of a combination** at step t is defined as the cost to move the robots to the cells of the combination. This cost is computed with an A-star planning algorithm, noted

Algorithm 2: One step of search common to `RandomWithContribution` and meta-heuristics (`TS`, `SA`)

Data: The set of m robots, the number of moving robots $\delta \in [1..m]$, a `ChooseAction` function $\in$ {`RandomAction`, `SimulatedAnnealing`, `TabuSearch`}
for $i \in [1, .., m]$ **do**
  Update o_i
`// 1- Choose which robots to move`
for $i \in [1, .., m]$ **do**
  Compute w_i
$weakRobots \leftarrow$ the δ robots with lowest w_i
`// 2- Choose an action`
for $i \in [1, .., \delta]$ **do**
  $c_i \leftarrow$ current cell of the robot $weakRobots[i]$
  $weakRobots[i]$ executes `ChooseAction`(c_i)

`AStarMove`(m, λ), that searches a path for each agent $\in [1..m]$ to each cell $\in \lambda$. The graph used by A-star expands each cell using its adjacent cells in the quadtree. The cost between two adjacent cells in the A-star graph is the distance a robot has to travel following the circular navigation topology. A cost penalty is added in case of radius change to put a penalty when the robots has to change of circle, given it requires the robot to rotate and to slow down. The heuristic to estimate the cost between two non-adjacent cells is the euclidean distance between the two positions of the cells.

4.2.2 Algorithm. Algorithm 3 details each step of the exhaustive combination algorithm. At the beginning of each step, the algorithm has the set of already visited leaf cells L, and the set of all possible combinations at this step Λ. Λ contains all the m-combinations of cells from the set L (also noted $\binom{L}{m}$). Λ is shared among all agents.

The exhaustive combination algorithm first searches the best combination λ^* in Λ. The **best combination** λ^* is the combination with the optimal quality and the lowest cost given the current position of the robots. So λ^* defines a joint position that gives an optimal observation on the scene. If λ^* is found, the robots move to the corresponding cells of λ^*. Otherwise, the robots explore the map to complete and update the set of combinations. The exploration consists in moving to unvisited cells in priority, otherwise trying to split and in last resort, moving to less visited cells.

4.2.3 Combination set updates and complexity. At the beginning of each step, Λ is the m-combinations of cells from the set L so it is composed of $\binom{L}{m}$ combinations. To find the best combination, algorithm 3 must look over all these combinations.

The set of combinations Λ is also updated and can increase at each step given the actions of the robots.

- If a robot moved to a cell never visited before, Λ is increased with $\binom{L}{m-1}$ new combinations formed with the new cell and all the combinations of the already visited cells L (the new cell is not considered in L here).

Algorithm 3: One step of `Combination` search

Data: The set of m robots, the set of combinations Λ, the set of unvisited leaf cells $Unvisited$, the set of leaf cells sorted by their number of visits $Lessvisited$

for $i \in [1, .., m]$ **do**
  Update o_i
$\lambda^* \leftarrow$ the best and closest combination $\in \Lambda$, given positions of the m robots
if λ^* exists **then**
  `AStarMove`(m, λ^*)`// Move the robots to` λ^*
else if $Unvisited \neq \emptyset$ **then**
  `AStarMove`$(p, Unvisited)$`// Move` $p \leq m$ `robots`
else if Agents with no assignment can split **then**
  Agents not assigned try to split
else if Agents with no assignment cannot split **then**
  `AStarMove`$(AgentsNotAssigned, Lessvisited)$
for $i \in [1, .., m]$ **do**
  Update Λ

- If a robot split a cell, all the combinations in Λ where the split cell is are modified to replace the split cell by one of its children (the one where the robot has chosen to move).

4.3 Local vs. Global approaches

Table 1 summarizes the four proposed algorithms given different characteristics. In each algorithm, robots are supposed at least to share the `obstacle map` characteristics. This means that they build a cooperative obstacle quadtree, i.e. they all have the same quadtree structure decomposed in cells, and obstacle occupancy associated to these cells (but observation data are not required). `Observation communication` characteristic requires the robots to communicate at each time step their current observation vectors. This is an assumption of all the algorithms that compute the marginal contribution to choose which robots to move at each step. `Observation map` is when the algorithm requires the robots to build a full cooperative quadtree, with all the cell data detailed in the Map Data (cf. subsection 3.2). Finally the `space search` characteristic defines if the robots are searching in a local space (adjacent cells around their current cells) or in a global search (all leaf cells of the quadtree).

This tables allows to underline which approaches rely on local vs. global information leading to different computational cost. Among them two extreme solutions are proposed : i) the random action selection, having a null cost in computation and memory, ii) the exhaustive search having an exponential complexity and requiring to store all observation vectors obtained in each cell. Between the two we examine how the selection of robots to move, depending on their marginal contribution, could improve dramatically the performances. Then we aim to study how meta-heuristic exploiting local information compares to the exhaustive search approach, while requiring few computational expenses.

Algorithm	`Obstacle Map`	`Observ. Comm.`	`Observ. Map`	`Space` search
random	x	-	-	-
random-contrib	x	x	-	local
meta-heuristic	x	x	x	local
exhaust.-search	x	x	x	global

Table 1: Characteristics of the four compared algorithms. x means that the characteristic is required.

5 COMPARISON OF PERFORMANCE AND COST

5.1 Simulator

To perform experiments, we developed a simulator that allows to run a large quantity of experiments in order to test the validity of our approach. **A main feature of our simulator is that its main parts (data structures, e.g. cell, quadtree, decision algorithms, interface) are used both for the simulation and for the control of our robots during real experiments.** For the simulation part, specific modules have been developed to simulate in the most accurate and realistic way key features of real mobile robots, environments and scene. A video presenting the simulator interface, the exploration and incremental mapping with simulated and real robots can be found at https://liris.cnrs.fr/crome/wiki/doku.php?id=videoaamas2018.

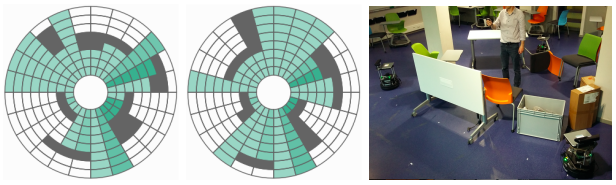
We assume that **robots' motion** around the scene is perfect, i.e. robots can move along circles without trajectory errors, and robots are equipped with sensors allowing them to remotely detect nearby obstacles. **Communications** between robots are also supposed to be instant and errorless.

To **simulate a scene**, we use **real data** obtained from real human captures with Kinect RGB-D sensor. These captures were made from different human poses (standing, sitting, crouching), with or without occlusions[8] (cf. fig. 3). For each scene, multiple captures were made from a set of point of views along a circle centred on the scene. Skeleton data obtained from OpenNI and NITE skeletal tracking library [7] have been imported in the simulator to generate the observation vectors[9] for each sector.

To **generate an environment** in the simulator, one can choose where to put obstacles or to randomly distribute them. Concerning observations from each cell, it is possible to choose random ones or observations obtained from a human pose generated from real data. Then we use a *ray tracing* [2] technique starting from the scene to assign to each cell of a same sector a common observation until reaching obstacles. Cells behind an obstacle have a null observation.

[8] Occlusions are not in the navigation space, i.e. they are in the space between the scene and the most inner circle of the navigation model.
[9] *i.e.* skeleton information composed of 15 body joints.

Figure 3: Photos of real human captures of different scenes representing human pose *standing* with various occluded scenes.



(a) Env. A. (b) Env. B. (c) Real environment.

Figure 4: (a) - (b):Two different environments used in the simulator. (c): Real environment with three robots around a human scene (*reading phone*) and a cluttered environment.

Once the reference environment has been generated, it is used in all runs. Fig. 4 shows two reference environments containing different obstacles (black cells). They also show cells from which the scene is visible (green cells) and not visible (white cells). Different shades of green indicate different local qualities of observation: the greener the cell, the better is the local observation. Environments we have used are designed in such way that it's not possible for one single robot to find a cell from which it can see the full joint observation.

To **simulate noise in camera sensor and occlusions due to other robots**, we add noise to the perception from a cell. A noise parameter β defines the probability for each value of an observation vector to be flipped (compared to the reference values). Robots occlusion is simulated as a noise inversely proportional to inter-robot distance. This operation is computed each time a robot asks for an observation over the environment. Thereby, the perceived observation from cells of a same sector may vary.

5.2 Experimental setting

We perform our experiments in two different environments, presented in fig. 4, using 3 robots and 3 initial cells. Simulation of the scene is done using real data captured from different human poses. Table 2 provides some information concerning the state space of each environment and the number of optimal joint positions for each pose, illustrating their complexity. In the following, common parameters used are the number of agents $m = 3$, the number of moving agents at each time step $\delta = 2$, the size of the tabu list is 5, the temperature for SA algorithm is set once to 0.6 at the beginning

	Env. A	Env. B
# Targets	15	
# Circles	8	
# Sectors	24	
# Leaf cells	192	
# Obstacles	27	
# Possible joint positions	1,661,853	1,616,586

	# Optimal joint positions
Env. A Pose *seated*	90,800 (5.46%)
Env. B Pose *seated*	146,235 (9.05%)
Env. A Pose *seated with boxes*	237,510 (14.29%)
Env. B Pose *standing with boxes*	465,158 (28.77%)
Env. A Pose *standing with chair*	500,319 (30.1%)

Table 2: Characterictics of env. A and B with 3 robots. Possible joint positions are all the combinations of non-obstacle, leaf and non-leaf cells. This number is not the same, as non-obstacle non-leaf cells are not the same in A and B.

of each experiment, and decreases at each step according to $\alpha = 0.01$, but is never reset until the experiment ends.

5.3 Performance comparison

First we plot the current joint quality found by the robots at each step with the different algorithms in Figure 5. It shows that with the `Random` algorithm, the current joint quality fluctuates a lot, contrary to the other algorithms. This illustrates that the heuristic that consists in moving only the robots with the lowest marginal contributions at each step is a good solution to maintain a stable view on the scene. Indeed, this heuristic is used with all algorithms except the `Random`. Maintaining a stable view on the scene is particularly interesting to detect potential changes in the scene, as in case of a dynamic scene.

Second we compute for each algorithms, the distance that the robots have to travel before covering optimally the scene, i.e. being on a joint position with the best possible joint quality. We do not count the number of algorithm steps, because one time step of the algorithms can last for variable time in real experiments. Indeed with random algorithms or meta-heuristics, robots can only move to their adjacent cells at each time step, so the distance travelled by each robot is relatively small and the duration of the steps is roughly of the same order. But with the `Combination` algorithm, each robot can move to any cells of the map at each step. And this move is done by following the circular topology (cf. §3.1). So one step in the `Combination` algorithm can last much longer than one step with the algorithms where robots are moving only to their adjacent cells. This is why we use the distance instead of the number of steps to fairly compare all the algorithms in simulation. This is equivalent to compare

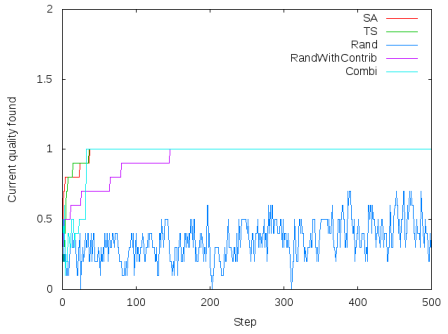


Figure 5: Current joint quality vs. step number with $m = 3$ robots, over 10 experiments. The environment is Env. A and the pose is *standing with occlusions*.

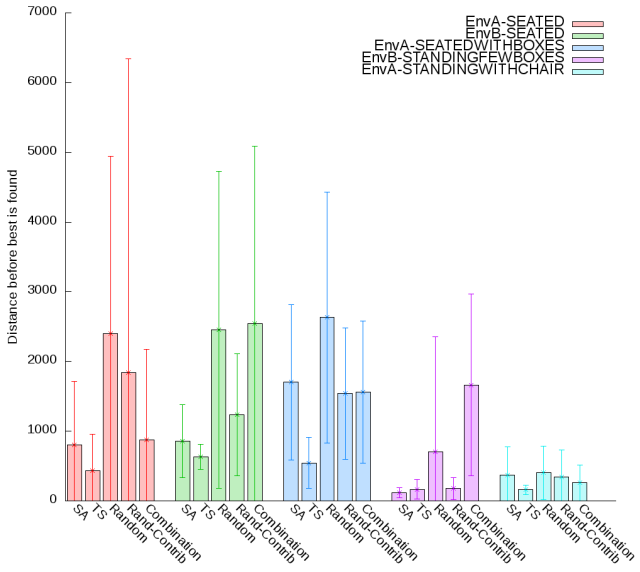


Figure 6: Mean distance (meter) and standard deviation travelled by the robots before finding the best possible joint quality for the first time (50 expe.).

the time taken by real robots before covering optimally the scene in case of real experiments.

Fig. 6 shows this distance for different environments and human poses. We can see that in all cases, `TS` meta-heuristic is the fastest in term of distance to cover optimally the scene. `Combination` algorithm shows various results. Indeed it needs to explore all the unvisited cells before being able to split. In occluded environments, the best joint position can require several split, which increases the number of cells to potentially explore before finding a combination with the best quality. These results highlight the importance of the use of the marginal contribution to the search process. They also show that strategies based on local information and meta-heuristic obtain better results than the exhaustive exploration / combination algorithm, especially considering the time to find a best joint quality. This time is an important factor in our context where we want to observe human scene that could be dynamic and so avoid long processes of exploration.

5.4 Real robot experiments

We performed experiments using 3 Turtlebot2 robots equipped with a RGB-D camera (Kinect) for the skeleton tracking, a low-cost 360° and 4 meters laser rangefinder (RP-Lidar) for local metric mapping and navigation, a netbook with Ubuntu/ROS connected to the mobile base and to the rangefinder for the decision and navigation part, and an Intel NUC mini-PC with Windows connected to the Kinect for the human observation part. ROS *gmapping*-package is used as particle filter SLAM algorithm. The decision algorithms (cf. §4) are embedded in ROS nodes. They request skeleton data to a server running on the NUC. Data exchanged between robots use TCP/IP socket between netbooks.

These experiments were realized with the same modules (data structures (e.g. cell, quadtree), decision algorithms, interface) as the one used in the simulator. The additional components used with real robots are: (i) an hybrid mapping mixing high-level data of the quadtree map and low-level data of the metric map; (ii) a communication module between the robots, (ii) a distributed architecture for the algorithms presented in §4; this architecture relies on asynchronous communication and topological quadtree maps cooperatively constructed by the robots. We tested `SA` and `TS` strategies and obtained similar results than with the simulator. Videos presenting these experiments are available at https://liris.cnrs.fr/crome/wiki/doku.php?id=videoaamas2018.

6 CONCLUSION

We formally defined the generic task of the multi-robot observation of a human activity in an unknown cluttered environment as a simultaneous problem of coverage and mapping. The objective is to optimize the robot joint-observations of a human scene while exploring the environment. We proposed an incremental mapping based on a quadtree structure and a concentric navigation topology allowing to manage the state space complexity of the task. Then we proposed and evaluated different strategies showing that considering each agent marginal contribution is essential to the search process, while approaches based on local information and meta-heuristic optimizations obtained better results than the exhaustive exploration/combination algorithm.

As a perspective, we plan to extend our work to dynamic scenes, i.e. when the person is doing a sequence of activities. In this context, the robots will have to detect the change and adapt their positions at each new activity. This refers to a coverage problem of multiple **moving** targets. To perform this fast adaptation, keeping some map information about obstacles and occluded cells could be helpful. Another perspective is to extend our approach to soft confidence values for body joints. This would require to redefine observation and marginal contribution in a probabilistic framework.

REFERENCES

[1] Ehsan Adeli Mosabbeb, Kaamran Raahemifar, and Mahmood Fathy. 2013. Multi-View Human Activity Recognition in Distributed Camera Sensor Networks. Sensors 13, 7 (2013), 8750. https://doi.org/10.3390/s130708750

[2] Arthur Appel. 1968. Some Techniques for Shading Machine Renderings of Solids. In Proc. of Spring Joint Computer Conf. 37–45. https://doi.org/10.1145/1468075.1468082

[3] Antoine Bautin, Philippe Lucidarme, Rémy Guyonneau, Olivier Simonin, Sebastien Lagrange, Nicolas Delanoue, and François Charpillet. 2013. Cart-O-matic project : autonomous and collaborative multi-robot localization, exploration and mapping. In 5th IROS Workshop on Planning, Perception and Navigation for Intelligent Vehicles. 210–215.

[4] A. Giusti, J. Nagi, L. Gambardella, and G. A. Di Caro. 2012. Cooperative Sensing and Recognition by a Swarm of Mobile Robots. In Proceedings of the 25th IROS. 551–558. https://doi.org/10.1109/IROS.2012.6385982

[5] Fred Glover and Manuel Laguna. 1997. Tabu Search. Kluwer Academic.

[6] Sergiu Hart. 2008. Shapley value. In The New Palgrave Dictionary of Economics, Steven N. Durlauf and Lawrence E. Blume (Eds.). https://doi.org/10.1057/9780230226203.1517

[7] PrimeSense Inc. 2011. Prime Sensor NITE Algorithms 1.5.

[8] Miguel Julia, Arturo Gil, and Oscar Reinoso. 2012. A comparison of path planning strategies for autonomous exploration and mapping of unknown environments. Autonomous Robots 33, 4 (2012), 427–444. https://doi.org/10.1007/s10514-012-9298-8

[9] A. Khan, B. Rinner, and A. Cavallaro. 2016. Cooperative Robots to Observe Moving Targets: Review. IEEE Transactions on Cybernetics PP, 99 (2016), 1–12.

[10] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. 1983. Optimization by Simulated Annealing. Science 220, 4598 (1983), 671–680. https://doi.org/10.1126/science.220.4598.671

[11] Gerhard K. Kraetzschmar, Guillem Pagès Gassull, Klaus Uhl, Guillem Pagès, and Gassull Klaus Uhl. 2004. Probabilistic Quadtrees for Variable-Resolution Mapping of Large Environments. In Eds., Proceedings of the 5th IFAC/EURON.

[12] Laëtitia Matignon, Laurent Jeanpierre, and Abdel-Illah Mouaddib. 2012. Distributed value functions for the coordination of decentralized decision makers. In International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2012, Valencia, Spain, June 4-8, 2012 (3 Volumes). 1209–1210.

[13] Lynne E. Parker. 2002. Distributed Algorithms for Multi-Robot Observation of Multiple Moving Targets. Auton. Robots 3:12 (2002), 231–255.

[14] Reid Simmons, David Apfelbaum, Wolfram Burgard, Dieter Fox, Mark Moors, and et al. 2000. Coordination for multi-robot exploration and mapping. In Proceedings of National Conference on Artificial Intelligence (AAAI). AAAI.

[15] Matthijs T. J. Spaan, Tiago S. Veiga, and Pedro U. Lima. 2010. Active Cooperative Perception in Network Robot Systems Using POMDPs. In Proc. of IROS. 4800–4805. https://doi.org/10.1109/IROS.2010.5648856

[16] Xiaogang Wang. 2013. Intelligent Multi-camera Video Surveillance: A Review. Pattern Recogn. Lett. 34, 1 (Jan. 2013), 3–19. https://doi.org/10.1016/j.patrec.2012.07.005