Autonomous Robots manuscript No.
(will be inserted by the editor)

Searching and Tracking People with Cooperative Mobile Robots

Alex Goldhoorn[1] · **Anaís Garrell**[1] · **René Alquézar**[1] · **Alberto Sanfeliu**[1]

Received: date / Accepted: date

Abstract Social robots should be able to search and track people in order to help them. In this paper we present two different techniques for coordinated multi-robot teams for searching and tracking people. A probability map (belief) of a target person location is maintained, and to initialize and update it, two methods were implemented and tested: one based on a reinforcement learning algorithm and the other based on a particle filter. The person is tracked if visible, otherwise an exploration is done by making a balance, for each candidate location, between the belief, the distance, and whether close locations are explored by other robots of the team. The validation of the approach was accomplished throughout an extensive set of simulations using up to five agents and a large amount of dynamic obstacles; furthermore, over three hours of real-life experiments with two robots searching and tracking were recorded and analysed.

Keywords Multi-Robot coordination · Urban Robotics · Search-and-Track · Decentralized Coordination

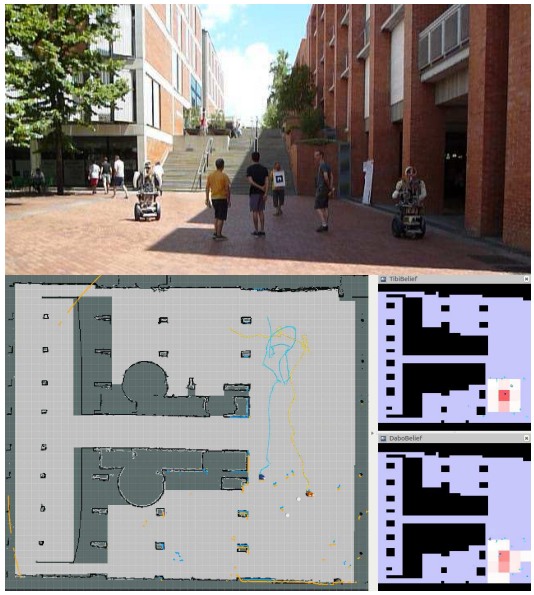

Fig. 1 The robots search and track the person (in the back wearing a tag to recognize him) while other people are walking around obstructing the robots' vision. In the lower map left, the localization of the robots can be seen (orange and blue robots), and at the right, the probability maps of the person's location of both robots are shown.

1 Introduction

Searching and tracking are important behaviors for a mobile service robot, for example to assist people, to search and

Work partially supported by the Spanish Ministry of Science and Innovation under project Rob-In-Coop (DPI2013-42458-P) and EU FP7 project ARCAS (INFSO-ICT-287617).

A. Goldhoorn
E-mail: agoldhoorn@iri.upc.edu; alex@goldhoorn.net
A. Garrell
E-mail: agarrell@iri.upc.edu
R. Alquézar
E-mail: ralqueza@iri.upc.edu
A. Sanfeliu
E-mail: sanfeliu@iri.upc.edu

rescue (Marconi et al. 2012; Sheh et al. 2016) or search for objects (Ferrein and Steinbauer 2016).

The method can be applied to searching objects, but here we focus on searching and tracking of a person. Even though searching might be evident for humans, for robots it is not, since it requires exploring; handling noisy sensors that also can give false positives or false negatives; coping with dynamic obstacles, such as other people walking in front of the robot; and in the case of multiple agents, coordination to do an efficient search.

In previous studies (Goldhoorn et al. 2014), we described and evaluated different methods for searching and tracking a person in urban settings, using a single humanoid service robot. These methods made use of the online search algorithm *Partially Observable Monte-Carlo Planning* (POMCP; Silver and Veness (2010)), which, in contrast to other previous approaches, can plan under uncertainty, in large continuous state space and in real-time.

In this paper, we present two multi-robot approaches (the *Multi-agent HB-CR-POMCP Explorer* and *Multi-agent HB-PF Explorer*) that can search and track one person, using one or more agents (robots) that cooperate by communicating their observations and most probable locations of the person. Internally, the agents use a probability map of the person's location, for which we tried two different methods: the *Highest Belief Continuous Real-time Partially Observable Monte-Carlo Planning* (HB-CR-POMCP) and the *Highest Belief Particle Filter* (HB-PF) method, for which the observations of all agents are used. Thereafter, the most probable locations are marked and sent to all agents, such that each agent can choose the best location to explore. In the case of a visible person, the agents track the person while updating the probability map, since the observation could be a false positive. The methods are able to cope with noisy sensors, false negative detections, and, for a short time, false positive detections. Furthermore, the methods are able to search and track with only one agent when no communication is available. We improve the person location probability map by also using dynamic obstacles—such as other people walking around—and a probabilistic visibility check.

Finally, the validation of the approach was accomplished by an extensive set of simulations and a large amount of real experiments in an urban campus environment with dynamic obstacles, using our mobile social robots Tibi and Dabo (Garrell and Sanfeliu 2012), see Fig. 1.

In the remainder of the paper, we start by introducing the related work of cooperative robotic search in Section 2, after which a global overview of the proposed approach is given. Next, we explain first the methods used to update the belief (Section 4), then, in Section 5, how the goals are selected. In Section 6, the experimental setup is explained, after which the simulations (Section 7) and real-life experiments (Section 8) are shown, and we end with some conclusions in Section 9.

[1] Institut de Robòtica i Informàtica Industrial (CSIC-UPC)
Llorens Artigas 4-6, 08028 Barcelona, Spain.

2 Related Work

The task of either tracking or searching by a mobile robot has been studied previously, but in few times both are combined in one method, like in ours; furthermore, we extensively tested the methods in real-life experiments. A simplification of the real world problem of finding people is the hide-and-seek game, where there are one or more agents searching, and one or more hiding. Hide-and-seek and pursuit-evasion (Chung et al. 2011) are well-known games which have been used in a large amount of—mostly theoretical—works to test and compare planning algorithms. The hide-and-seek game also requires a high number of cognitive functions such as: search, navigation, coordination, anticipation and planning (Johansson and Balkenius 2005). In (Goldhoorn et al. 2013b,a), we focused on the hide-and-seek game in discrete time and space, using MOMDPs (Mixed Observable Markovian Decision Processes; Ong et al. (2010)) to search for a person. In (Goldhoorn et al. 2014) we extended it to play in real-time with continuous states (Continuous Real-time POMCP).

Surveillance also requires tracking, which was done by (Capitan et al. 2016) with Unmanned Aerial Vehicles (UAVs). They did tracking of multiple targets and they assumed them to move independently. Their method used an MOMDP, with as discrete state space the combination of the target's and UAVs' locations, and as actions four movement directions and one staying on the same position. For each behavior (target to track), a different POMDP policy is learned, and the behavior is selected using an auction method. The policies were learned for a reduced state space (a single target for a single UAV), since it is intractable for the combined state space. They did simulations and experiments with small UAVs in a small artificial environment.

Volkhardt and Gross (2013) used a service robot to detect and find people in a scenario with three rooms. To detect a person they looked at the legs, face, body-shape and motion, which is a more realistic recognition method than the use of an artificial tag; however, the tag allowed us to do experiments in a large outdoor environment. Their search was guided by a list of predefined guide points, and they assumed there to be only one person. Challenges like RoboCup (Ferrein and Steinbauer 2016) try to promote research in Artificial Intelligence and robotics by organizing competitions in different fields, first they started with soccer competitions, but they also have a search-and-rescue track (Sheh et al. 2016). Tracking has been extensively discussed in the

SPENCER project, of which perception and tracking of people (Linder et al. 2016) was one of the goals. In (Linder et al. 2016), authors compare different algorithms to track people with a mobile robot in a busy airport terminal. They found a method that uses the nearest neighbour and an extended Kalman filter to work best. In the SHERPA project (Marconi et al. 2012), they focused on search-and-rescue activities using a mixed group of ground and aerial vehicles, and humans.

Many works make use of *Particle Filters* for tracking (Thrun et al. 2005), since it is a fast algorithm, its complexity mainly depends on the number of particles and it allows for any distribution, unlike a Kalman filter, for example, which requires a Gaussian distribution. In (Montemerlo et al. 2002), the authors tracked a large distribution of person locations, conditioned upon a smaller distribution of robot poses over time. Glas et al. (2015) introduced a tracking algorithm using individual particle filters to track multiple entities with multiple robots. Oyama et al. (2013) presented a robot that tracks visitors' positions in a museum guide tour. In contrast to the previous approaches, our method makes use of the cooperation of several robots to not only track, but also search, therefore, we do not need an initial observation of the person.

In (Cui et al. 2008), combined laser scanners and video images to track multiple people are introduced, to overcome the limitations of visual trackers; it first detected feet, and then the person was searched in the video image; however, they used fixed locations for the laser scanners and camera. (Lian et al. 2015) did tracking of a person in a dynamic environment by trying to maximize the visibility of the target. First, they used a laser range finder and an extended Kalman filter, then a look-ahead algorithm (DWA*) to follow the target and avoid obstacles at the same time. None of these methods explicitly search for a person, nor do they mention keeping track of people when they have been hidden for a long time. Ahmad and Lima (2013) tracked a spherical object (ball for the RoboCup soccer challenge) with a team of robots. They used a particle filter and they shared the observation, observation confidence and the localization confidence. The confidences were used as weights to update the particle filter. The method is similar to our particle filter method, but we use a fixed observation confidence. We, however, also share the most probable locations and we do an explicit search of the person by exploring the most probable locations. Their experiments were done on a RoboCup soccer field with four mobile robots.

In (Luber et al. 2011), a combined multi-hypothesis tracking method is presented; it uses a Kalman Filter with an online detector that has as input color and depth data. Their experiments were in a crowded indoor environment using three Microsoft Kinect sensors. Also (Choi et al. 2011) used Kinect sensors to track people, using a variant of a particle filter to keep track of several targets. Authors in (Brscic et al. 2013) did tracking of people in large public environments, where they used multiple 3D range sensors mounted at above the human height, to reduce occlusions. Each tracked person was assigned an identifier if he/she had been visible as a new cluster during several steps; if the dispersion of the cluster got too high, the person got deleted, and was recovered when the person got detected close to the identifier. Experiments in a shopping centre showed good results. Although several 3D sensors, such as the Kinect, give good detection results indoors, they do not work well outdoors. We have detected the person, by combining the person legs detection with a Lidar and the detection of an artificial tag to recognize a specific person.

To do tracking with multiple agents, the combination of decentralized techniques and particle filters has led to *Distributed Particle Filters* (DPFs) (Sheng et al. 2005; Hlinka et al. 2013), which have been used by, for example (Vázquez and Míguez 2017). These works focus on tracking of one or more people with a *Wireless Sensor Network*, where they use a large number of connected sensors. Our method on the contrary, works with one or more mobile robots, without depending on a pre-installed fixed sensor network.

Researchers (Sheng et al. 2005) use DPFs to localize and track several targets with a wireless sensor network, and in order to reduce the information sent between nodes, they use a low dimension Gaussian Mixture Model (GMM). They compared methods that work separately, in sensor groups, and hierarchically. They worked with a previously proposed *Centralized Particle Filter* (CPF) tracking algorithm in which the posterior distribution is updated based on all measurements, however, Sheng et al. do the update in groups of sensors. Tests were done in simulations on an area of $100 \times 100\ \mathrm{m}^2$ with 25 fixed sensors and two targets to track. Vázquez and Míguez (2017) presented a DPF that uses the median posterior probability in order to combine efficiently local Bayesian estimators; in simulations they showed that their method is more robust.

Multi robot teams are also used, as in (Xu et al. 2013), who tracked a visible target with a decentralized robot team, thereby learning the utility models of the robots and negotiating with the other robots. They used an Information Filter (IF), which is a variant of a Kalman filter, with as goal minimizing the uncertainty and optimizing the information obtained by the robots. Experiments were done with two Segway RPM robots, one with 360°, and another with 180° vision.

Hollinger et al. (2010) presented an online decentralized multi-agent search algorithm that creates a path to find a person on a graph. It generates a scheduler to calculate a search plan for multiple agents. They tried to optimize the path based on an adversarial and non-adversarial person model. Whereas we use a probabilistic approach to keep track of

the probable locations of the person, Hollinger et al. kept track of a list of *contaminated* nodes (areas that not yet have been checked, or where the person could have returned to), thereby assuring the person to be found. However, to assure a person to be found, a minimum number of search agents are necessary, which depend on the map configuration. Furthermore, their maps were converted to graphs, where in each node they assumed full vision, whereas we use a vision probability based on distance and obstacles. Next, they did not handle on-line changes in the environment, where we do take into account dynamic obstacles. And finally, they only searched for the person, but our methods also track the person. In (Hollinger et al. 2015), they focused on data fusion between the agents, and they kept track of the probability of the person being in each of the vertices. When there is communication, they take into account the other agents' paths, otherwise, after reconnection the beliefs are fused. They showed two simulated experiments, one in a map like in the previously mentioned paper, the second in an underwater sea environment, where communications disturbance is a real problem. In our method we do not send the complete belief, but we send the observations of the agents, and after having locally updated the probability map (belief), the most probable locations are sent to the other agents.

Charrow et al. (2013) used a team of robots with range sensors to localize a fixed radio source. For each robot a measurement of the distance to the radio source was taken, which was used as input to a particle filter. The robots had a reading, but with noise, depending on how many obstacles are between it and the target. They used the entropy to optimize the control strategy for all the robots, reducing the uncertainty of the estimation of the target location. They did experiments with real robots on two environments up to 40 m $\times$ 35 m. In our work, we use sensors that requires the target to be in the field of view, and within a certain distance in order to recognize the person, which is a more realistic situation, even though we make use of a marker to recognize the person.

3 Overview of the Approach

This section gives a global overview of the proposed approach, thereby also mentioning the constraints.

3.1 System Architecture

In this work we present a method for multiple mobile robots to search and track a person autonomously and cooperatively, which at the same time, allows the robots to operate individually when there is no communication between them, or when only a single robot is available. The method uses a probability map (*belief*) to represent the probability of the location of the person. It also requires a *map* of the environment on which each robot should be able to localize itself. This map was created beforehand using the odometry and the laser range detectors, as explained in Section 6.3.

The diagram in Fig. 2 gives an overview of the approach presented in this work, which consists of four phases. In the first phase, for *Robot Localization*, Odometry and Lidar are used. *People Detection* is accomplished by a Lidar, and markers are used to recognise the person (Section 6.2). The detected people are also taken into account in the *Update Belief* algorithm (as dynamic obstacles). Finally, the *Observation Filter* makes sure that the locations of the person and robot are legal (i.e. within the map and not in an obstacle), by taking the closest most probable location. Note that the *Person Localization* module can be replaced by any other detector, in order to search and track a specific object for example.

Together with the observations of the other agents, the belief is updated in the *Person Localization* phase, for which two algorithms were tried: the *Multi-agent HB-CR-POMCP Explorer* (based on Goldhoorn et al. (2014)), in which each robot uses the probability map of the CR-POMCP to search and track a person; and the *Multi-agent HB-PF Explorer*, which makes use of a particle filter.

In the third phase, the *belief* and observations are used to decide on the locations where the robots should search for the person—which we will call *goals*. If a robot detects the person, then *Tracking* is carried out, otherwise the *Exploration* method is applied. The latter chooses the goals for each robot from the list of highest belief points of all the robots. The goals are chosen by taking into account the probability, the distance to the goal and whether another agent already has a goal close to it.

Finally, the robot's path planner (Section 6.3) plans and executes the path to the chosen goal.

3.2 Problem Constraints and Model Assumptions

This subsection describes the assumptions made for the model in simulation, and the limitations we came across while testing our model in real-life scenarios. There are at least two types of problem constraints: the first derive from the robot's perception and actuators; the second are the result of human behavioural reactions to the robot's instructions. The effects of these limitations on our study and the model assumptions are summarized below:

- For safety reasons the robots were not allowed to go faster than around 1 m/s.
- Also for safety, the robots are kept at a minimum distance of the person, other persons, other robots and any detected obstacles.

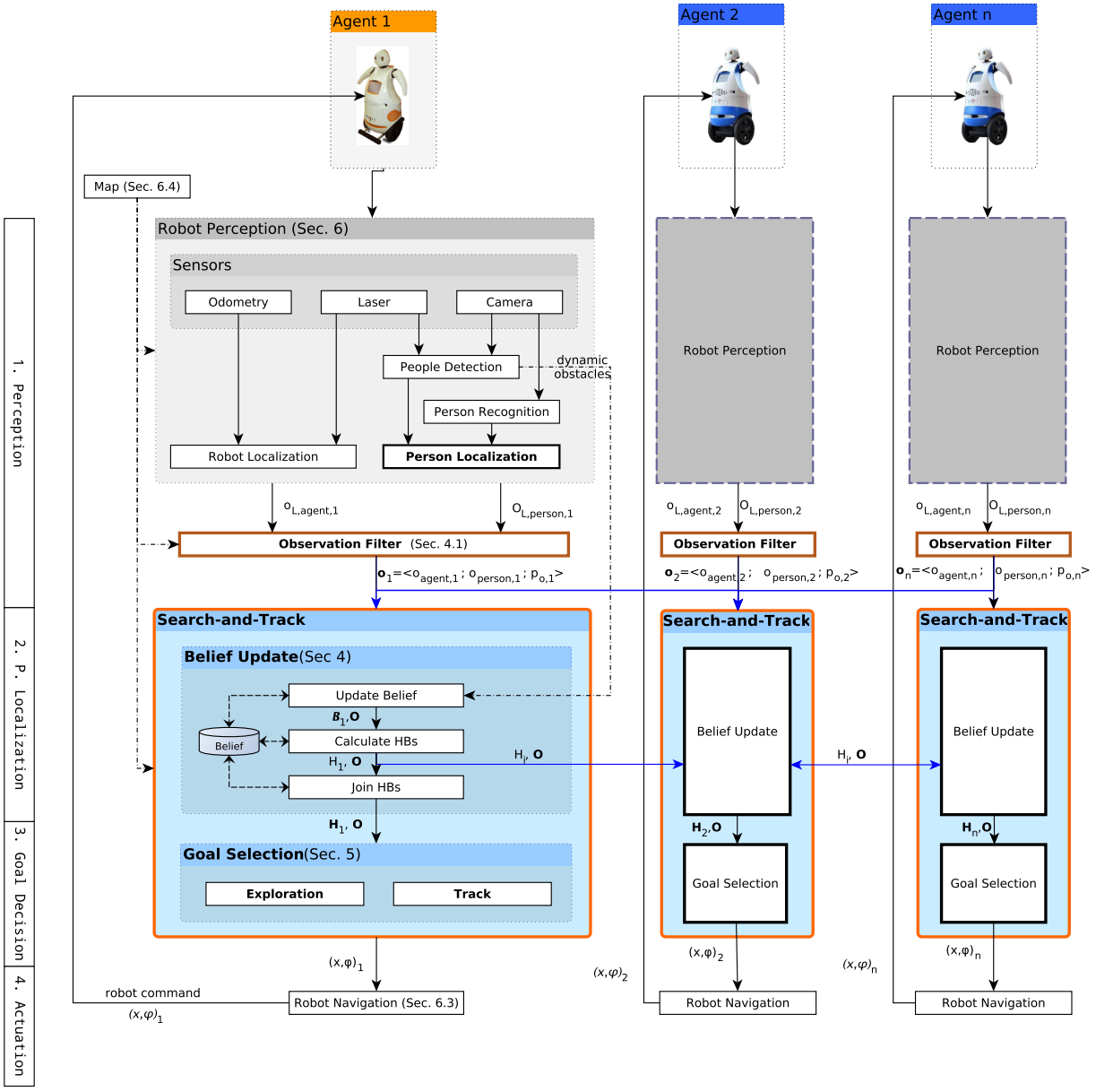

Fig. 2 The schema shows the complete search-and-track approach for n agents, with the same diagram for each agent and the communicated data, where the diagram of the first agent is shown in detail. At the left the phases of the search-and-track method are shown. The blocks are algorithms or groups of algorithms, the orange bold lined blocks were created by us. The black arrows show how the output of one algorithm is used as input for another, and the blue arrows show the communication between the agents. The sections in which the items are discussed are shown between parenthesis.

- The person being followed is asked not to walk too fast (i.e. less than 1 m/s).
- The map of the environment, i.e. location of obstacles (walls, doors, objects, etc.) has to be known beforehand in order to plan and predict, therefore, we use a map of the environment.
- There are no methods to recognise a person robustly outdoors from a large distance and from any perspective. For that reason and because our research is focused on searching and tracking, we make use of an artificial marker (Section 6.1) to recognise the person.
- A 360° view is assumed, to reduce the state space, and therefore, simplify the planning (not taking into account the orientation and field of view).
- For simplicity, the static obstacles are assumed to occlude everything and do not allow to let anyone pass.
- Dynamic obstacles occlude like static obstacles, but they can move and are not present in the map. In simulation

we do allow the agents to collide with them, to prevent the simulation becoming too complex.
- In simulation we also allow agents to collide with each other, also to prevent a too complex simulator.

4 Belief Update

In this work, we use the probability of the location of the person on a known map, the probability map is called *belief*. Two methods were tried to create the belief: the first is based on *Partially Observable Monte-Carlo Planning* (POMCP; Silver and Veness (2010)); the second is based on *Particle Filters* (Thrun et al. 2005). Both methods use particles to represent the belief, but the first uses the belief update method of the POMCP, whereas the second method uses the standard Particle Filter update.

To decide where the robots should search for the person, the points with the highest probability, the *highest belief* points, are calculated (Section 4.4). It also explained how this information is combined with the other agents' *highest belief* points.

Finally, we explain how the search goals for the robots are chosen, by selecting the points with the high probability, the *highest belief* points. And we explain how this information is combined with the other agents' *highest belief* points. After the highest belief points are calculated, they are used to send the robots to their goals, as Section 5 explains. Section 5 explains how the highest belief points are used to calculate the goals of the robots.

4.1 Preliminary

The input of the belief update algorithms is the list of observations $\boldsymbol{O} = \{\boldsymbol{o}_1, \boldsymbol{o}_2, \cdots, \boldsymbol{o}_n\}$ of the different agents, as shown in Fig. 2. Each observation $\boldsymbol{o}_i = \langle o_{\text{agent},i}, o_{\text{person},i}, p_{o,i} \rangle$ contains the location of the robot ($o_{\text{agent},i}$), the observation of the person ($o_{\text{person},i}$, which can be *empty*) and a correctness probability $p_{o,i}$. The correctness probability indicates how much a person detector can be trusted and was estimated based on experimental results.

Ideally, the observations are sent and received synchronously, however, in the real world this was not the case. In simulation there was no delay of the observations of the other agents, but in the real-life experiments there was. Therefore, we set a time limit for the observation to be used in the other robots. If the observation was too old (we have used the limit of 3 s), it was not used. Although asynchronous or delayed observation messages do result in a difference in the beliefs of the agents, in a later step they send each other's highest belief points.

The methods make use of a known map on which the location of the person is estimated, and the location of other visible dynamic obstacles (such as people) is used. The vision of the agent is limited by static and dynamic obstacles, the probability of seeing position s_1 from position s_2 is given by:

$$P_{\text{vis}}(s_1, s_2) = \begin{cases} 0, & \text{if } \textsc{ray}(s_1, s_2) \text{ not free} \\ p_{\text{v,max}}, & \text{else if } d < d_{\text{v,max}} \\ \max(0, p_{\text{v,max}} & \\ \quad -\alpha_{\text{vis}}(d - d_{\text{v,max}})), & \text{otherwise} \end{cases} \quad (1)$$

where RAY is the raytrace function, which makes use of the discrete map; $d = \|s_1 - s_2\|$; $p_{\text{v,max}}$ is the maximum visibility probability; $d_{\text{v,max}}$ is the distance until which it has a maximum visibility probability; and α_{vis} is the slope with which the probability function reduces. The parameter values were tuned based on real world data and are shown in Table 1. The time complexity of the raytrace algorithm is linear with the number of cells, but in practice it is constant, since we cache the results.

For the multi robot case, the probabilities are combined to calculate the probability of seeing position s_2 from any position $s_1 \in S$:

$$\bar{P}_{\text{vis}}(S, s_2) = 1 - \prod_{s_1 \in S} (1 - P_{\text{vis}}(s_1, s_2)) \quad (2)$$

4.2 Multi-agent HB-CR-POMCP Explorer

POMCP is a reinforcement learning method that is based on the *Partially Observable Markovian Decision Process* (POMDP; Pineau et al. (2003); Kurniawati et al. (2008)), and uses Monte-Carlo simulations instead of finding the optimal value function—using value iteration (Pineau et al. 2003) for example. POMDPs have states, observations, actions, rewards and two probability functions. States, in our case, are the locations of the person and robot (both continuous); observations are the robot's location and the observed location of the person (discrete for the policy tree), or *hidden* if not visible; and the reward is the negative distance to the person, i.e. higher when closer to the person. There were nine actions: moving one step in eight directions and staying at the same position. For POMDPs two probability functions are defined; one function defines the probability of going from one state to another with a specific action; the other defines the probability of an observation given a state and action. For the POMCP, instead of using the entire probability matrix, a POMDP simulator $(s', o, r) = \mathscr{G}(s, a)$ is used, which returns a new state s', observation o, and reward r, based on a current state s, and action a. This results in a computational complexity that mainly depends on the number of simulations n_{sim}.

Instead of knowing the current state, a belief is maintained, which is the probability of being in any of the states.

In POMCPs, the belief is maintained as a list of n_{belief} possible states, instead of a probability of each state. Note that for the POMDPs, normally, states are discrete which lets the belief be stored per state, whereas for the POMCP the belief can be continuous, as in the *Continuous Real-Time POMCP* (CR-POMCP; Goldhoorn et al. (2014)). The system is initialized with a belief b_0, which in our problem is based on the initial observation o_0 of the agent. When the person is visible initially, all belief is located there (with some added noise), otherwise, it is spread among the not visible locations. For this we use the map and the probability of visibility (2).

The POMCP algorithm (Silver and Veness 2010) generates a policy tree—which indicates what action to take to reach the highest reward. The policy is created by doing a large number of n_{sim} simulations, using the POMDP simulator $\mathscr{G}$. In (Goldhoorn et al. 2014), the policy tree is generated real-time and in each step it indicates the best action to take. The next belief is calculated during the policy generation phase, and when the action has been executed and the new observation has been obtained, the new policy tree root is chosen from the tree itself (using the observation and action). The new tree root's belief is extended if necessary, to contain at least n_{belief} points.

From the experiments with the robot, however, it was found that using the actions of the POMCP policy resulted in an inefficient movement behavior (Goldhoorn et al. 2014). Therefore, we decided to only use the belief of the POMCP, from which the highest belief points were chosen as search locations for the robot.

To cope with sensor noise and actuator noise, Gaussian noise was added in the POMDP simulator with a standard deviation of σ_{person} for the person's movement, and σ_{robot} for the robot's movement. Also false negative and false positive observations were simulated with probabilities $p_{\text{false\_neg}}$ and $p_{\text{false\_pos}}$ respectively. The list of parameters and their values is shown in Table 1, and more details about the *HB-CR-POMCP* can be found in (Goldhoorn et al. 2014).

4.2.1 Multi-agent POMCP

In the initialization phase the belief is calculated based on the visibility, like explained earlier. For the multi-robot case all the observations each have their own probability ($p_{o,i}$), which was added to take into account the accuracy and trustworthiness of the sensors of specific robots. To generate the initial belief, n_{belief} states are generated by randomly picking observations $o = \langle o_{\text{agent}}, o_{\text{person}}, p_o \rangle \in \boldsymbol{O}$ with probability p_o and some Gaussian noise added (with standard deviation σ_{person}). If the person is not visible in that observation, a random position is chosen, which is not visible to any agent.

After having generated the belief, the POMCP policy tree is created by doing n_{sim} simulations. Then, the best action is chosen from the policy tree, and when it has been executed, the *belief update* is done.

Algorithm 1 The belief consistency check function, with as input the state s and the observation vector $\boldsymbol{O}$. The function RANDP generates a random value between 0 and 1.

```
1:  function CONSISTENCYCHECK(s, O)
2:      isVisible = false
3:      for o ∈ O do
4:          if not o_person is hidden then
5:              if ||s_person − o_person|| > d_cons then
6:                  return false
7:              else
8:                  isVisible = true
9:              end if
10:         end if
11:     end for
12:     p = P̄_vis({o_agent | o ∈ O}, s_person)
13:     if isVisible then
14:         if RANDP( ) > p then return false
15:         end if
16:     else
17:         if RANDP( ) ≤ p then return false
18:         end if
19:     end if
20:     return true
21: end function
```

Before the belief update, all the observations $\boldsymbol{O}$ are received from all agents, as can be seen in Fig. 2. The belief is updated with the observation o, which includes only the information of the own agent, because including other agents' positions would make the policy tree grow very wide, and thereby resulting in an exponential growth of the policy search.

The belief is first updated by choosing the new belief root from the policy tree. Second, the states in the new belief are checked for consistency with all the observations. States that were not found to be consistent were removed from the belief. Algorithm 1 shows the consistency check, which is done for each state in the belief ($s \in \mathscr{B}$), using the observations of all agents $\boldsymbol{O}$. First, it checks if the observed person locations are either *hidden* or close to the belief state s, then (2) is used to calculate the probability that the person location of the state should be visible to any of the agents. Finally, a random function is used to decide the consistency, taking into account the visibility probability.

As third step, states are added to the belief until it has n_{belief} states. Each new state s is randomly chosen from $\boldsymbol{O}$, and if s_{person} is *hidden*, then it is set to a random location where the person is not visible to any of the agents' locations.

4.3 Multi-agent HB-PF Explorer

The way we use the CR-POMCP algorithm to track the person resembles the way particle filters (Thrun et al. 2001) are used to track an agent. In the CR-POMCP algorithm,

Algorithm 2 A basic Particle Filter.

1: $\bar{S}_t = S_t = \emptyset$
2: **for** $i = 1$ to $n_{\text{particles}}$ **do**
3: sample $s_t^i \sim p(s_t|s_{t-1}^i)$
4: $s_{t,w}^i = p(o|s_t^i)$
5: $\bar{S}_t = \bar{S}_t \cup \{s_t^i\}$
6: **end for**
7: **for** $i = 1$ to $n_{\text{particles}}$ **do**
8: sample $s_t^i \in \bar{S}_t$ with probability $s_{t,w}^i$
9: $S_t = S_t \cup \{s_t^i\}$
10: **end for**

Algorithm 3 The update step of the search-and-track particle filter.

1: **function** UPDATE($\boldsymbol{O}$,S,$n_{\text{particles}}$)
2: $\forall_{s \in S} : s_w = \min_{o \in O}(w(s,o))$ ▷ get minimum weight
3: $\forall_{s \in S} : s_w = s_w / \sum_{k \in S} s_w$ ▷ normalize
4: $\bar{S} = \emptyset$
5: **for** $i = 1$ to $n_{\text{particles}}$ **do**
6: sample from $\bar{s} \in S$ with probability $\bar{s}_w$
7: $\bar{S} = \bar{S} \cup \{\bar{s}\}$
8: **end for**
9: $S = \bar{S}$
10: **end function**

the belief contains a list of possible locations, which can be compared to the particles in a *Particle Filter*.

Particle filters and Kalman filters have been applied in many works to localize a robot (Montemerlo et al. 2002; Glas et al. 2015; Oyama et al. 2013), however, they require an observation. In our problem we do not always observe the position of the person, so the Kalman filter and particle filter only could do a prediction step, which might be sufficient for very short time periods. For longer periods, the prediction will get invalid, since the person is not always going straight. We have chosen to use particle filters, since they have been proven to work well for tracking, are able to represent different types of distributions, are easy to adapt to our problem and have a low computational complexity.

4.3.1 Particle Filters

Particle filters are used to estimate the posterior of the state, based on observations. Here the state is the position of the person, and we focus on searching and tracking the person We do not use the same method to track the robot's position, such as done in (Montemerlo et al. 2002) for example. A standard particle filter (Thrun et al. 2005) estimates the current state based on all its observations: $p(s_t|o_{0:t})$. Algorithm 2 shows that there are different steps, first a prediction step (line 3), then the weight is calculated based on the observation (line 4), and finally, resampling is done (line 8) based on the weight. The complexity of this method is linear with the number of particles.

4.3.2 Adaptations for Search-and-Track

Tracking algorithms normally start with an initial particle distribution close to the measured location of the person, in our case however, we do not always know the person's location. Therefore, when the person is not visible to us, the $n_{\text{particles}}$ particles are spread on the map over the areas which are not visible to the agent(s), as explained in Section 4.2.1.

The prediction step is a Gaussian movement in a random direction: $s_t = s_{t-1} + \mathcal{N}(1, \sigma_{\text{person}})[\cos\theta, \sin\theta]^T$, where σ_{person} is the standard deviation, and θ a random direction.

The update step is shown in Algorithm 3, where first the weight is calculated using the following equation:

$$w(s,o) = \begin{cases} 0, & \text{if } \neg \text{ISVALID}(s) \\ e^{-|o_{\text{person}} - s|^2 / \sigma_b^2}, & \text{else if } \neg(o_{\text{pers.}} = \textit{hidden}) \\ w_{\text{cons}}, & \text{else if } P_{\text{vis}}(o,s) = 0 \\ w_{\text{inc}}(1 - P_{\text{vis}}(o,s)), & \text{otherwise} \end{cases} \tag{3}$$

where ISVALID indicates whether the state is within the map, and not in an obstacle; σ_b can be used to tune the area over which the weight is spread, we set it to 1.0. P_{vis} (1) gives the probability of being visible, therefore, if there is no observation (i.e. *hidden*), and the particle position is consistent with the observation, then we assign a constant weight w_{cons} (0.01). Otherwise, a lower weight $w_{\text{inc}} \ll w_{\text{cons}}$ is given (we set it to 0.001).

In Algorithm 3, the weights are calculated for each observation $o \in \boldsymbol{O}$ in line 2, where they are aggregated taking the *minimum*. Using the *maximum* would cause that inconsistencies are only detected if all agents detect them as inconsistent (and thus give it a low score, see (3)), otherwise agents that are far enough not to see the particle, score it higher ($w_{\text{cons}} \gg w_{\text{inc}}$). In this case a *minimum* should work better, because it remarks the inconsistency of the particle. Finally, an *average* can also work, since it takes into account all the observations, but it will eliminate inconsistent particles slower. To verify if both these methods work we have done simulations, taking the *minimum* and the *average* score.

4.4 Highest Belief

Finally, to decide a goal for the seeker agent, we can use the average position of the particles, which makes sense when only tracking is done, since the particles will be close to the target and normally distributed. In our case however, this will not be the case when the agent has to search the person, which can be any not visible location. Therefore, we make

Algorithm 4 The explorer finds the goals g_i for all agents i using the score function (4).

1: **for all** $h \in \boldsymbol{H}$ **do**
2: $\quad U_h = 1$
3: **end for**
4: **for all** $i \in$ Agents **do**
5: $\quad g_i = \arg\max_{h \in \boldsymbol{H}}$ EXPL_SCORE$(s_{\text{person},i}, h)$
6: $\quad$ **for all** $h \in \boldsymbol{H}$ **do**
7: $\quad\quad U_h = U_h - P(\text{DIST}(h, g_i))$
8: $\quad$ **end for**
9: **end for**

use of a 2-dimensional histogram to find the highest probable location. This method, the *Highest Belief*, was proposed in (Goldhoorn et al. 2014).

The 2D histogram is made by counting the number of particles per cell, and dividing them by the number of particles ($n_{\text{particles}}$ or n_{belief}) to get the probability of the person being there, see Fig. 7 for an example of the histogram. The size of the cells of the histogram should be large enough to increase the stability, but small enough to have enough precision. In our experiments we have used cells of 3.2 m $\times$ 3.2 m.

Next, the n_{hp} highest belief points H_i are selected, and are sent to the other agents. Each $h \in H_i$ contains a position h_{pos}, and a belief h_b. The received highest probability points (H_i) of all agents and of the agent itself are joined by summing the beliefs for each highest probability point, and thereby generating the set of all highest belief points $\boldsymbol{H}$.

5 Goal Selection

After the belief is updated and the *Highest Belief* points are created, received, and joined, the *Goal Decision* phase (Fig. 2) starts, where the robots either tracks the person or explore the most probable locations. If the person is visible and the observations are consistent, then the agents follows the person side-by-side (Garrell et al. (2013); *Tracking* in Fig. 2). Otherwise, the agents explore the joined highest belief locations $\boldsymbol{H}$, as shown in Algorithm 4, which is based on the work of (Burgard et al. 2005). Each agent calculates the goals for all agents, using the joined highest belief points $\boldsymbol{H}$. A score is calculated for each highest belief points $h \in \boldsymbol{H}$ per agent location s:

$$\text{EXPL\_SCORE}(s,h) = w_u U_h + w_d \frac{\text{DIST}(s,h)}{d_{\max}} + w_b \frac{b_h}{b_{\max}} \quad (4)$$

where U_h is a utility function for highest belief point h, s the agent's position, and DIST calculates the shortest path distance. The second and third term are normalized by the maximum distance $d_{\max}$ and maximum belief $b_{\max}$, with respect to the list of potential target locations $\boldsymbol{H}$. The utility U_h (Burgard et al. 2005) is initialized with 1 (line 1 of Algorithm 4), and updated before searching the goal of the other robot (line 4), where g_i is the already assigned goal to agent i, and:

$$P(d) = \begin{cases} 1.0 - \frac{d}{d_{\text{max\_range}}}, & \text{if } d < d_{\text{max\_range}} \\ 0, & \text{otherwise} \end{cases} \quad (5)$$

with $d_{\text{max\_range}}$ being the range within which we want to reduce the chance of other agents' goals being chosen. The terms of (4) are weighted by w_u, w_d, and w_b, and the values we found to work well are: $w_u = 0.4$, $w_d = 0.4$, and $w_b = 0.2$.

The order in which the agents are assigned the goals is important, since the assignment of a goal h to an agent reduces U_h, and therefore reduces the probability of other agents being assigned this goal. We chose to assign the agent with the highest sum of probabilities ($\sum_{h \in H_p} b_h$) a goal first, and the lowest last. Most importantly, the order should be consistent for all agents such that all agents calculate the same goals, assuming they have received all highest belief points H_i. With this method it can occur that an agent a_1 is assigned a goal g_1 that is further away than a goal g_2 assigned to a_2, because the latest was assigned firstly. Note that finding the closest goals $g \in G$, such that the sum of the distances with the agents $a \in A$ is minimum: $\min_{a \in A} \sum_{g \in G} \text{DIST}(a,g)$ has a complexity of $O(\|A\|!)$. Therefore, we approximate it, by re-iterating over the calculated goals and assign the closest goals, in the same order (i.e. on sum of the highest belief). It can be seen that for agents from which no positions have been received, no goals are calculated.

Finally, to prevent changing the goal too often, the goal is only changed every t_{update} time, or when the person is visible.

The method explained in this section is only guaranteed to give the same search goals for all agents if they receive all the highest belief points of all agents synchronously. If not all highest belief points are received, the resulting search goal positions may be close to each other, which results in a less efficient search.

6 Experimental Setup

In this section the used robot and environmental maps are explained.

6.1 The Robots

For the experiments we have used our mobile service robots, Tibi and Dabo, which have been created during the URUS project (Sanfeliu et al. 2010) to interact with people in urban pedestrian areas. They are based on a two-wheeled Segway

RMP200 platform, which can work as an inverted pendulum in constant balancing, can rotate on the spot (nonholonomic), and they have wheel encoders providing odometry and inclinometers providing pitch and roll data. To perceive the environment they are equipped with two Hokuyo UTM-30LX 2D laser range sensors, used to detect obstacles and people, giving scans over a local horizontal plane at 40 cm above the ground, facing forward and backward. The lasers have a long detection range of 30 m, and a field of view of 270°, which is limited to 180° for each of the lasers because of the carcass of the robot. Additionally, a distance of about 45 cm between the front and rear laser causes a blind zone. As video camera Dabo uses a PointGrey Ladybug 2 360° camera, located on the top of its head; whereas Tibi uses a Bumblebee 2 stereo camera at the front and two Flea 2 cameras at the back, which in total cover much less than 360°, and therefore has less vision.

As social robots, Tibi and Dabo are meant to interact with people, and to perform this, they have: a touchscreen, speaker, movable arms and head, and LED illuminated face expressions. Power is supplied by two sets of batteries, one for the Segway platform and one for the computers and sensors, giving about a five hours of full working autonomy. Two onboard computers (Intel Core 2 Quad CPU @ 2.66 and 3.00 GHz with 4 GB RAM) manage all the running processes and sensor signals. As operating system the systems run Ubuntu 14.04 with ROS (Robot Operating System), a middleware.

6.2 People Recognition

To detect people, and recognize the target person, both range laser and vision have been combined. A boosting leg detector (Arras et al. 2007) provides the position of potential people in the scene, using the horizontal front and rear range laser sensors. False positives are reduced by filtering out detections that are close to, or inside a known obstacle. A Multiple Hypothesis Tracking For Multiple Targets (Blackman 2004) keeps the trail of the people and assigns them identifiers.

A people detection algorithm is not enough, because we also have to recognize the person we are looking for. A robust method is to use AR Markers (Augmented Reality Markers) (Amor-Martinez et al. 2014), which were worn by the person, see Fig. 1. The AR algorithm gives an estimation of the pose with respect to the camera. We used an improved version of this Pose Estimation algorithm of Amor-Martinez et al, which in combination with previous local window binarization makes the method more robust to outdoors lighting issues. On Dabo we use the Ladybug 360° camera (which internally has five cameras) to detect a tag from any direction, and on Tibi we use four cameras with smaller angles of view. The AR detection algorithm is run on one computer for all cameras and ran on average at 4 Hz. False positive detections of the AR Markers are reduced by accepting only detections close to a laser detection; which, as side-effect, generates some false negatives.

6.3 Robot Mapping and Navigation

Prior to the experiments, a map was generated by the robot using the range lasers, with the ROS package GMapping, which implements OpenSlam's GMapping. This is a highly efficient Rao-Blackwellized particle filer that learns grid maps from laser range data (Grisetti et al. 2007). Although this method can be used for localization and mapping, we did not want to use it during the experiments, because it also can mark persons as being obstacles if they stand still for too long. Instead, we used the Adaptive Monte Carlo Localization (AMCL) approach, also available as ROS package, for localization. This method uses a particle filter to track the pose of a robot against a known map (Arulampalam et al. 2002).

The robot moved through the environment using a set of navigation algorithms provided by ROS. A Dijkstra global planner uses the previously generated map to calculate the shortest path. To avoid dynamic obstacles, a local Trajectory Roll Out planner is used, which generates and scores trajectories over a costmap that is updated with range laser data. The input of the navigation algorithm is the desired goal coordinates and orientation.

6.4 Environments and Maps

Experiments were conducted in the Barcelona Robot Lab (BRL), *Telecos Square* of the North Campus of the Universitat Politècnica de Catalunya (UPC), Barcelona, Spain, see Figures 1 and 7. The area has a size of 60 m $\times$ 55 m (about 1400 m^2 free space), and contains a square with trees, a terrace and a covered area with several columns.

7 Simulations

This section explains the setup of the simulations and a detailed analysis of the results.

7.1 Setup

The maps contain discrete cells that either are free or contain a (static) obstacle. The agents cannot see through obstacles (static or dynamic), and they can not pass through static obstacles. To make the simulation not too complex, the agents can pass through each other and across dynamic obstacles.

Although the map contains cells, coordinates of the agents are continuous. And for each iteration the agents do a step of 1 cell distance (also in diagonal, thus not $\sqrt{2}$) in the direction of its goal. The simulations do not include neither acceleration, nor friction, nor collision, for simplicity.

A ray tracing algorithm is used in simulation to detect visibility due to obstacles. In contrast to our previous work (Goldhoorn et al. 2014) we limit the visibility also based on the distance using a probability function (1) and the parameters (shown in Table 1) were tuned based on real world data.

A crowded environment was simulated by adding a group of 10 or 100 people (dynamic obstacles) to the scene, who reduce the robot's visibility, but they did not block the agents' paths. The movements of the simulated people (including the person to be found) were semi-random, they were given a random goal to which they navigated to, using a shortest path algorithm; a new random goal was assigned to them when the goal was reached.

More than 40,000 experiments were done, repeating each of the conditions at least 250 times. For each run of simulations the robot's start position, and the person's start and path were generated randomly. To make the comparison as fair as possible, the same positions were used for all the algorithms and conditions, such that the initial state and the paths of the person and the dynamic obstacles were the same.

7.2 Simulation Goals

In the simulations the two belief update algorithms were tested: the *Multi-agent HB-CR-POMCP Explorer* and the *Multi-agent HB-PF Explorer*. For the latter, two fusing methods were tried for the observations of the different agents in the update phase: the *average* and the *minimum*. As an upper line we added a best-case algorithm, the *See All Follower*, which is a follower that always knows the location of the person, independent of the distance or any obstacles being between the seeker and the person.

The goals of the simulations were to see how well the presented search-and-track methods worked for multiple agents and under different circumstances. Here we limited the tests to adding up to 100 dynamic obstacles and using up to five seekers that either had communication or had not. We split the simulations in two types: in *searching* and *tracking*. For searching, the person should be found as fast as possible, and for tracking, the agent should be close to the person as long as possible while seeing him/her. In all cases the *See All Follower* should work best, since it always sees the person.

The *searching* simulations were started with the person being hidden to all the seekers and without moving. The simulations ended when either a robot reached the person at a distance of 1 cell (0.8 m in the used map), or 2000 steps were reached. The simulations were measured using the time it took for at least one seeker to see and to be next to the person. The *tracking* simulations were done with the person being visible to one or more of the seekers and continued for 500 time steps. Another measurement was the distance between the seeker and the person, hereby taking the lowest distance over all of the seekers.

Furthermore, a measurement of the belief (probability map) of the person ε_b has been introduced, which indicates the error of the person's location in the belief with respect to the real location, which can only be calculated in the simulation. The value ε_b is a weighted distance between the person's location in the probability map and in reality:

$$\varepsilon_b = \sum_{x \in A} b_x \|x - p\| \tag{6}$$

where A is the discrete map, x represents a grid cell, b_x is the probability of cell x and p is the real (continuous) location of the person.

7.3 Algorithm Parameter Values

The values of the parameters used in the simulations and real experiments, which were explained in Sections 4 and 5, are shown in Table 1. The *HB-CR-POMCP* method updated its belief every 3 s in the real experiments and every 3 iterations in the simulations; the other parameters for the HB-CR-POMCP algorithm are explained in more detail in (Goldhoorn et al. 2014). The parameters $p_{o,\text{Tibi}}$ and $p_{o,\text{Dabo}}$ indicate the trustworthiness of the sensors, and since the vision of Tibi was less than the 360° vision of Dabo, we gave it a lower probability. All the parameters were tuned first in simulation, and later while doing tests with the real robots.

7.4 Results

The results of the search simulations are shown in Fig. 3, where the average time (discrete steps) it took to find the person is visualized. The time is measured until one of the seekers found the person and is next to him/her. The influence of communication is shown in the rows and the effect of the number of dynamic obstacles is shown in the columns. Since none of the data were normal, we used the Wilcoxon ranksum test, 2-sided to compare the different conditions.

For all cases the *See All Follower* was significantly faster ($p < 0.001$) than any other algorithm, since it was always able to see everything. Fig. 3 shows that it took more than four times longer when using one seeker with the particle filter method. When using only one seeker, the particle filter was significantly faster than the CR-POMCP ($p < 0.001$). For the multi-agent simulations, the use of communication

Table 1 The parameters values used during the real experiments and the simulations.

Parameter	Value	Description
Common Parameters		
σ_{person}	0.2 m	standard deviation of Gaussian noise
$p_{\text{v,max}}$	0.85	maximum probability visibility (1)
α_{vis}	0.17	reduction factor (1)
$d_{\text{v,max}}$	3.0 m	maximum distance full visibility (1)
$p_{o,\text{Tibi}}$	0.3	trustworthiness of Tibi's observations
$p_{o,\text{Dabo}}$	0.7	trustworthiness of Dabo's observations
Multi-agent HB-CR-POMCP Explorer		
n_{sim}	2500	number of simulations
n_{belief}	2000	number of belief points
$p_{\text{false\_pos}}$	0.001	false positive probability
$p_{\text{false\_neg}}$	0.3	false negative probability
d_{cons}	0.7 m	consistency check distance Algorithm 1
Multi-agent HB-PF Explorer		
$n_{\text{particles}}$	2000	number of particles
σ_b	1.0	tune spread of particle weight (3)
w_{cons}	0.001	weight (3) when observation consistent
w_{inc}	0.0001	weight (3) when obs. inconsistent
Highest Belief		
cell size	3.2 m × 3.2 m	2D histogram cell size
n_{hb}	10	number of highest belief points
t_{update}	3 s / 3 steps	wait time to re-calculate goal
Goal Selection		
w_u	0.4	utility weight for explorer score (4)
w_d	0.4	distance weight for explorer score (4)
w_b	0.2	belief weight for explorer score (4)
$d_{\text{max\_range}}$	30 m	maximum range of influence score (5)

was also significantly better ($p < 0.05$), except for some cases with the *Multi-agent HB-CR-POMCP Explorer*. In most of the cases, the *Multi-agent HB-PF Explorer* was the fastest method, and in particular the version that used the *average*.

For the *track* phase we want the seeker to stay close to and have the person visible as long as possible. Fig. 4 shows the average time it took to find the person again after losing him/her. The *See All Follower* still was best, but between the tested methods there was no clear winner, nor did communication give an advantage for one or another method, which most probably was because the robots were close to the person already (see Fig. 4). The increasing number of robots reduced the recovery time significantly, however, we did not simulate robots blocking each other's path, which in the real world would have reduced the efficiency of having multiple robots in a small area.

The average distance between the person and the closest agent when tracking is shown in Fig. 5. The particle filter method resulted in lower distances, and also using communication resulted in lower tracking distances.

The belief error (6) was calculated for the algorithms that use a probability map of the location of the person. For the search simulations the overall average and standard deviation of the belief error were 25.4 ± 8.9 m when there was communication and 27.8 ± 7.5 m without. Fig. 6 shows the average belief error for the *track* phase. The lowest belief error for the search simulations with communication was with the *Multi-agent HB-PF Explorer* method, using the *minimum* weight combination. There was no clear difference in the other cases.

The influence of having more dynamic obstacles is not clear (i.e. no significant difference for most cases) in the search time (Fig. 3), because they only block the agents' vision and not the path, i.e. the robot can go through the dynamic obstacles. From Fig. 4 can be seen that 10 dynamic obstacles almost did not influence the recovery time, but 100 did. Because of the large surface (1400 m^2), having 10 people walking around randomly had a low probability of influencing the vision of the robot, whereas 100 had a much higher probability. Finally, the influence of dynamic obstacles can also be seen in the average distance to the person (Fig. 5) and the belief error (Fig. 6).

To summarize, we found that, as expected, the base line *See All Follower* was faster in searching, and it tracked the person during the longest time. For *searching* we found the *Multi-agent HB-PF Explorer* to be faster than the *Multi-agent HB-CR-POMCP Explorer* in most cases, and in general, there was an improvement when using communication. *Tracking* showed no statistical difference between the methods (except for the *See All Follower*) in recovery time; it only showed that having more seeker agents resulted in a better performance. For the distance to the person while tracking, the *Multi-agent HB-PF Explorer* showed slightly better results. As weight combination method for the *Multi-agent HB-PF Explorer* when searching, the *average* was found to be slightly faster, but the *minimum* resulted in a slightly lower belief error.

8 Real-life Experiments

The simulated experiments were done to know how well the different methods worked under different circumstances (dynamic obstacles and with several seeker agents). We however, also wanted to verify how well the method worked in real-life. Therefore, we used our robots Tibi and Dabo to verify the *Multi-agent HB-PF Explorer* method in a large environment, the UPC campus (Telecos Square). We tried the version that used the *minimum* scores when using the observations in the particle filter update phase.

Like in the simulations, for the *search* behaviour we measured the time to encounter the person (by the first robot), and for the *track* behavior, we measured the recovery time and average distance to the person. Since we did not have a ground truth available, we had to use the information obtained through the sensors of the robots and the videos, which show the behavior of the robots. This had as consequence that the distance to the person was only measured when the person was visible.

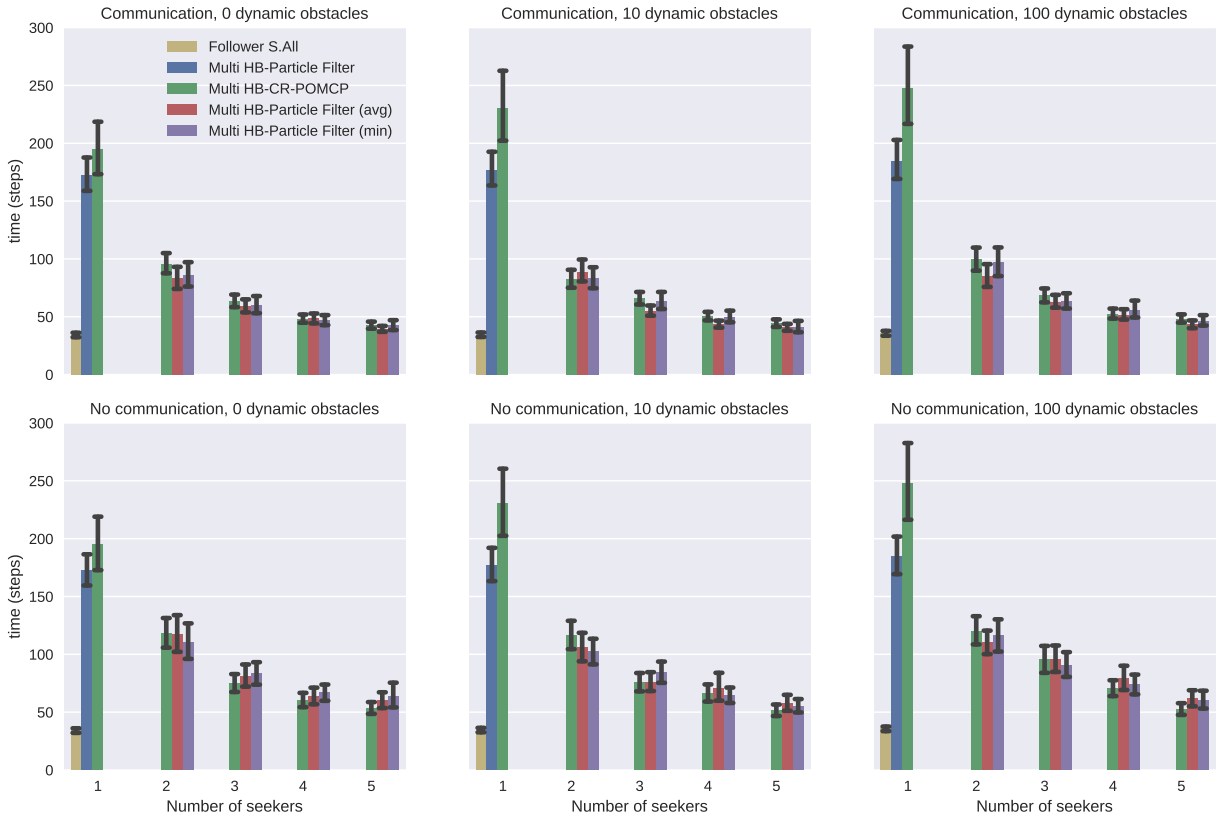

Fig. 3 The graphs show the average (and 95% confidence interval bars) of the time it took for one or more seekers to find and get close to the person. In the first row there is communication between the seekers, in the second there is not. In the columns the number of dynamic obstacles change. As a reference, the *See All Follower* took 34.7 ± 0.6 steps (mean $\pm$ standard error).

8.1 Analysis

Different types of experiments were done: *exploration* without a person, *searching and tracking*, and *tracking* only; they took several weeks of testing and experimenting, from which we obtained a total of about 3 hours of experimental data, and whereby the robots drove each a total distance of about 3 km. A few persons were used during the experiments in which they hid behind one of the obstacles, or just stood out in the open. The robots tried to follow the person at a distance of 1 m, and they always tried to maintain a minimum distance of 1 m to the other robot. The parameters used during the experiments are shown in Table 1.

Table 2 gives an overview of the different statistics of all the experiments. The distances shown were measured using the robot's sensors, i.e. the robot's movement, but also the person's moved distance, and therefore, is not complete, since the person was not visible the whole time. The *distance per robot* indicates the total distance covered on average by the robots during the experiments, the *measured dist. person* indicates the distance which was covered by the person, while the robot measured it. The *visibility* indicates the time the person was visible to a robot, the *time connected* indicates the time the robots were exchanging data. The *average distance to the person* is the distance between the robot and the person, measured when the person was visible. The *number of dynamic obstacles* are the average number of people which were visible simultaneously. The *average time found* is the time it took, on average, for a robot to find the person. Finally, the *average recovery time* is the time it took to find the person after having lost him/her.

In the next subsections we try to compare the results with the simulations, using the *time found* for the *search* experiments, and the *recovery time* and *average distance to the person*. However, in the real experiments the robot sometimes stopped or slowed down (due to obstacles, noisy signals or the low speed), therefore, the comparisons with the simulations should be done with the distance. Since the speed in the simulations was continuous (0.8 m per discrete time step), we can use this to calculate the distance covered, and then compare the distances to the *distance found* and *recovery distance*. Nevertheless, we should take into account that we can not do a statistical comparison of the results, since this would require many more experiments.

Fig. 7 shows two recordings taken during the experiments: the snapshots, the maps with the robot locations,

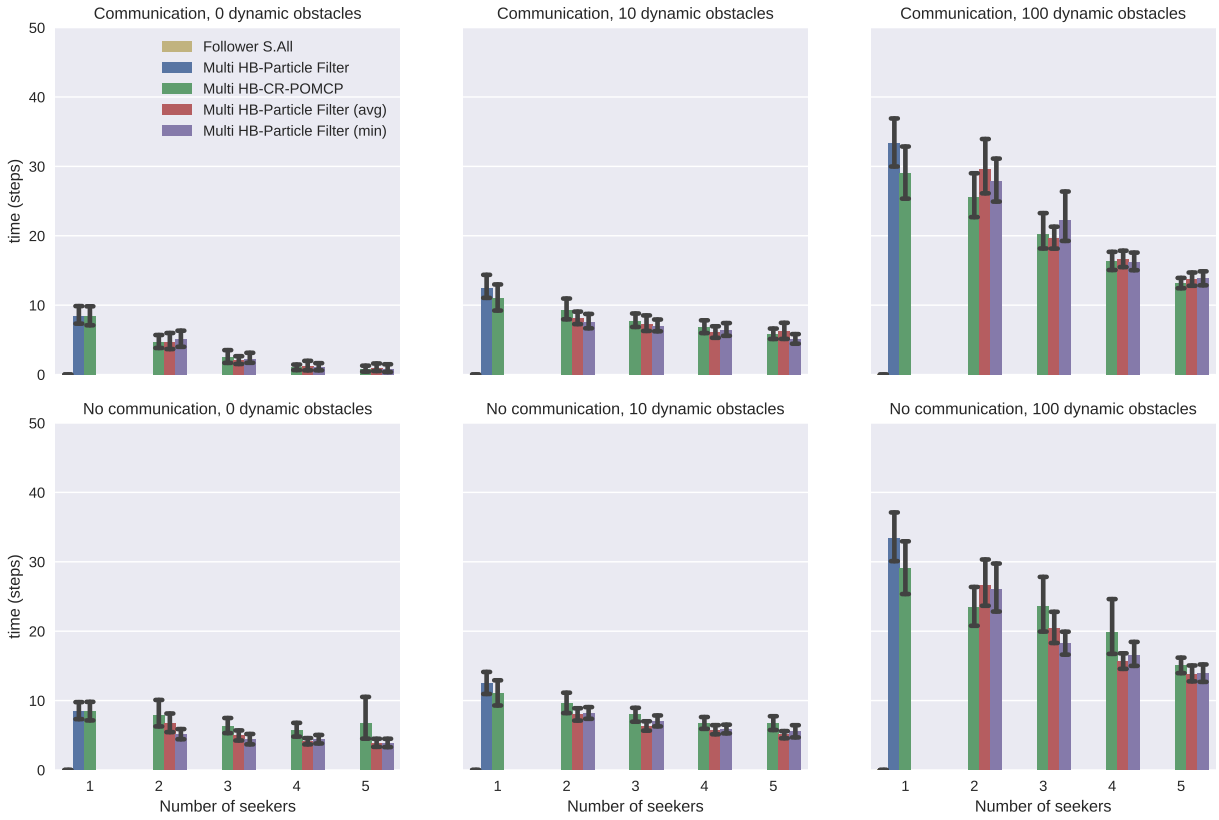

Fig. 4 The graphs show the average (and 95% confidence interval bars) time to discover the person after having lost him/her, due to (dynamic) obstacles for example.

Table 2 Summary of the data recorded during all the experiments. The averages (avg) are shown as *average*±*standard deviation*. *Measurements which include the person location were only available when the person was visible to a robot.

	Exploration	Search & T.	Tracking	Total
Distance per robot (km)	1.2	1.2	0.7	3.2
Measured dist. person* (km)	-	0.4	0.5	0.9
Total time (h)	1.1	1.2	0.9	3.2
Avg. visibility (%)	0	16.3	36.4	15.3
Avg. time connected (%)	95.0	79.5	85.8	86.6
Avg. distance to person (m) *	-	8.4±6.4	8.4±5.6	8.3±5.9
Avg. number dynamic obst.*	2.0±1.5	0.6±1.3	3.9±2.8	1.9±2.2
Avg. time found (s)	-	106.8±138.7	23.5±42.5	72.9±117.6
Avg. distance found (m)	-	69.3±74.0	6.2±13.6	27.3±53.3
Avg. time recovered (s)	-	19.6±39.0	12.0±28.3	15.3±33.6
Avg. distance recovered (m)	-	8.5±12.3	3.3±9.3	3.6±9.5

and the belief maps of both robots. The belief map shows that, when the person was detected, the localization was relatively precise (right), but when it was not detected for some time, the location probability is more spread (left). Further information and videos of the experiments can be found on: `http://www.iri.upc.edu/groups/lrobots/search-and-track/ar2016/`

First we will explain the three different kind of experiments done, followed by a short discussion.

8.1.1 Exploration Only

In these experiments we wanted to have a look at the search behavior, and therefore no person was present; this can be seen in Table 2, because there is no person distance. An exploration/search phase is shown in the left of Fig. 7, where none of the robots saw the person and both have a different belief. The experiments showed that the robots clearly explored the whole environment several times, because the belief slowly propagated to locations that were not visible to the robot.

8.1.2 Search-and-Track

In these experiments a person was present and the robots started not seeing him/her. The robots kept communicating the observations and therefore, could update their belief. They also explored in different directions looking for

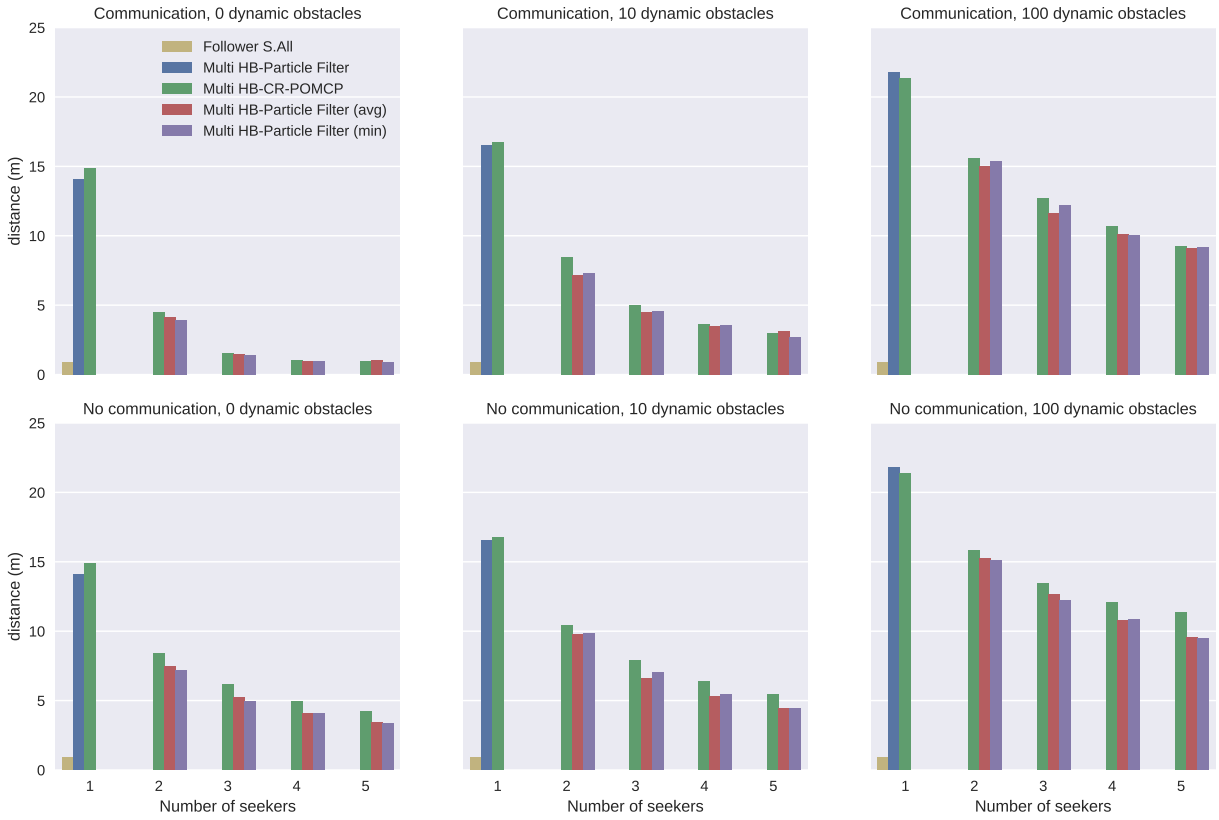

Fig. 5 The graphs show the average distance between the person and the closest seeker when following. The rows show communication or not, and in the columns the number of dynamic obstacles change. The *See All Follower* had the person always in sight and therefore was at a distance of about 0.88 m, i.e. the following distance.

the person. As soon as one robot saw the person, the other robot also went there. There were also situations where the person was lost, because he went faster than the robot, or because one robot temporarily failed; however, the belief of the working robot still helped to recover the person.

The distance covered by the robots until a person was found, was on average 69.3 ± 74.0 m, which is close to the distance covered in simulation, 67.5 ± 66.9 m (see Fig. 3 for the time with 0–10 dynamic obstacles, which was converted to distance). For the *tracking* part, the *recovery distance* is 8.5 ± 12.3 m, which is also close to the simulation's 5.1 ± 7.5 m (converted to distances, see Fig. 4). ehe *average distance to the person* shows a low value (8.4 m on average), because only measurements were taken when the person was detected by the robot.

8.1.3 Tracking

In the *tracking* experiments the robots started with the person being visible, and then followed him/her, but due to speed or (dynamic) obstacles they lost the person out of sight temporarily. Nonetheless, the person was found relatively quickly again, because he/she was tracked using the belief.

For some of the *tracking* experiments, the robots had to detect the person first, which took on average 23.5 s, but only 6.2 m, because the person was close. The *recovery distance* is 3.3 ± 9.3 m, which is also close to the values in simulation (5.1 ± 7.5 m). The *average distance to the person* was a bit higher, because the robot was relatively slow, and because having two robots tracking the person requires them to be at a minimum safe distance.

In the last experiment the robots searched for the person, which was behind or close to a group of people who occluded him/her, see Fig. 1. Since there were two robots, they had a higher probability of seeing the person, but when they did not see the person, the belief grew in all directions with a higher probability on areas where the robot probably would not see anything. Here, the dynamic obstacles (small light blue circles in the belief map) were taken into account, and the particles propagated behind fixed obstacles and dynamic obstacles (which was not done previously). Due to the low resolution of the belief map, there was also a belief at the location of the other people and the robot. Note that

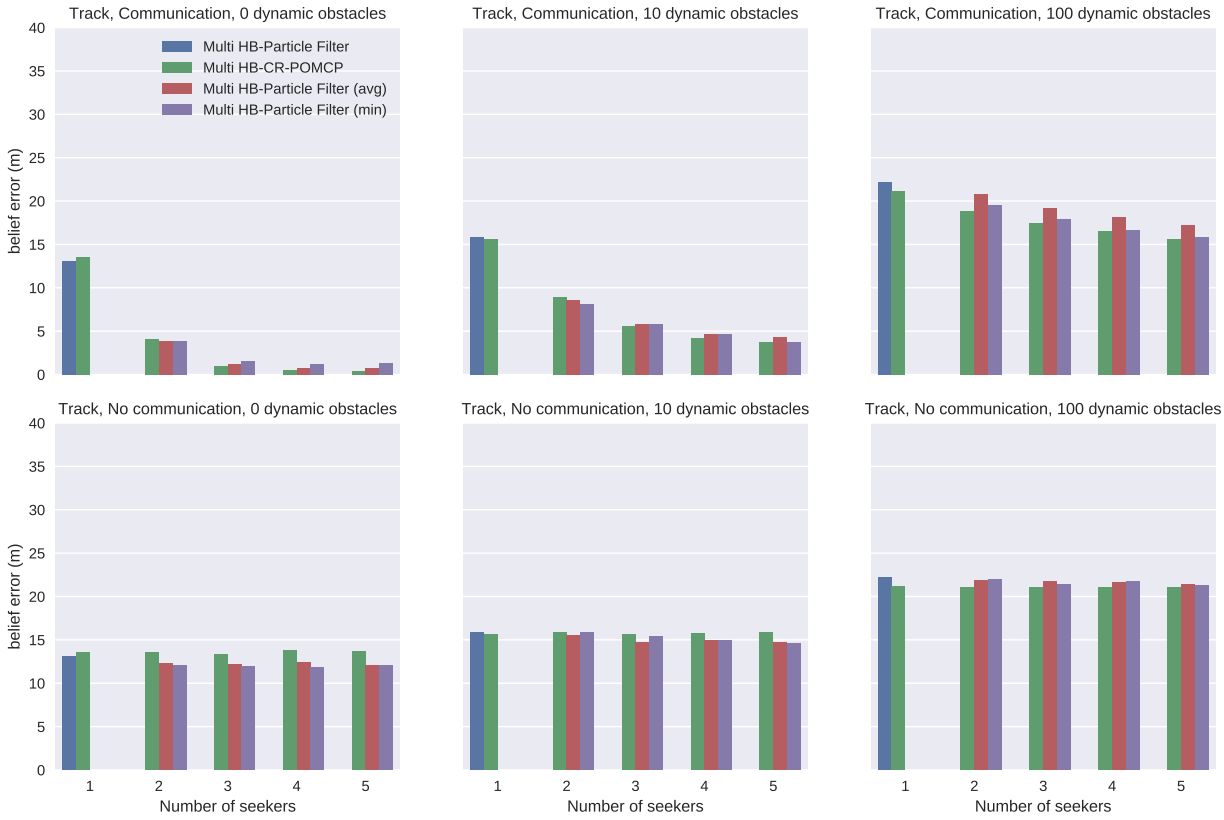

Fig. 6 The average belief error (using (6)) when following. The rows show communication or not, and the columns the number of dynamic obstacles. The *See All Follower* does not use a belief and is therefore not mentioned.

the low resolution of the map was chosen such that we could group enough particles to create a higher certainty.

8.2 Discussion

The experiments showed that the robots explored the whole environment, thereby taking into account the location of each other. And when tracking, it was also demonstrated that maintaining the belief continuously is important when the person gets out of sight. Furthermore, the robustness of the multi-agent method was shown in experiments where one robot suddenly stopped (because of a hardware or software problem). Then, the other robot recovered the person's position, since it had been receiving the person's location until the other robot stopped and it did not receive any information from the other robot. Therefore, using its own belief and observation, it only planned the next goal for itself.

False positive detections only occurred a few times, concentrating the belief slowly on that location, but—when the duration of the false positive was not longer than a few seconds—the belief expanded again, allowing the robots to continue searching. False negative detections simply delayed detecting the person.

Finally, we will discuss some issues with the methods while doing the experiments. First, the robots took the same path several times when they explored while this—according to a human point of view—might not be most efficient, since taking different paths allows them to explore more. Our exploration algorithms, however, do not take into account the path, only the goals are optimized such that the robots choose the closest most probable goal, which is not yet chosen by the other. To take the path into account, we should change the navigation algorithm, which might be complex when the number of seekers is high. Charrow et al. (2013) tried to optimise for maximal information and therefore, indirectly take the paths into account.

Second, the belief maps of Tibi and Dabo were not always equal, even though they received the same observations—if the communication worked—because there is a random factor in the propagation of the particles, which causes a different spread of the belief. When the seekers are without communication, they can only use their own observations and therefore, their beliefs will most likely be different. When they recover the communication they do not send historical information, and although this might be a useful feature, it can be a large amount of information if the amount of seekers is high. In (Hollinger et al. 2015), the

beliefs are fused by taking a weighted sum of the neighbors' beliefs.

The robots sometimes were not able to drive up or down the ramp due to the narrow passage and the inclined position, which made the horizontal lasers detect the floor as an object. In some cases this caused the planner to avoid the ramp and take a detour. In order to cope with ramps, a three dimensional map and navigation method should be used.

9 Conclusion

In this work, we have presented a unified method for searching and tracking a person using a group of mobile robots in a large continuous urban environment with dynamic obstacles. The observations are obtained from a leg detection algorithm that uses laser sensors and a marker detection algorithm in order to recognise the person. However, our method does not require a specific sensor type, but requires a location of the person or an empty observation—if not visible—as input; moreover, the observations of all other agents are used. At first, the belief of the person's location is maintained using either the *Multi-agent HB-CR-POMCP Explorer* or the *Multi-agent HB-PF Explorer*, then this belief is segmented in a histogram matrix to obtain the locations with the highest probability of the person being there. Thereafter, in the goal decision phase, the agents are either sent directly to the location of the person if he/she was visible, otherwise an exploration is done of the most probable locations.

Simulations were done in a large urban environment, part of a campus, with up to 100 dynamic obstacles moving around. For *searching*, in most cases, the *Multi-agent HB-PF Explorer* was fastest in finding the person, and in particular using the *average* weight, when using the observations of all agents. Also communication showed significant improvement for searching. For *tracking* we did not find any significant difference between the methods, neither when using communication. Furthermore, when looking at the tracking distance, the Particle Filter methodsegot closer to the person. And having multiple robots communicating, reduced the average tracking distance. Finally, the belief of the Particle Filter method was found to be closer to the real position.

The real experiments showed consistent results with the simulations and demonstrated it to be a pragmatic method to search and track a person in the real world with two robots. The search behavior showed an exploration over the field, whereby both robots were coordinating, and the communication between them also showed a more robust system, for example when one robot failed the other continued tracking the person quickly. The method was also shown to be a robust tracker when several people (dynamic obstacles) obstructed the vision of the robot temporarily, because they were able to find the person quickly again.

9.1 Future Work

The exploration can be improved by taking into account the path which the robots take such that they also explore the environment, like (Charrow et al. 2013) for example who try to maximize the mutual information of the agents.

In our experiments, the robots were able to communicate during most of the time, but when during some time the communication is not possible, the information of the other robots is not used to update the belief. To compensate this, the belief of each robot could be communicated such as in (Hollinger et al. 2015), but when the number of agents is high, the network bandwidth might be too high.

Finally, in order to analyse the effects of the communication delays and to verify the rest of the methods more simulations and real-life experiments should be done.

References

Ahmad A, Lima P (2013) Multi-robot cooperative spherical-object tracking in 3D space based on particle filters 61(10):1084 – 1093, selected Papers from the 5th European Conference on Mobile Robots (ECMR 2011)

Amor-Martinez A, Ruiz A, Moreno-Noguer F, Sanfeliu A (2014) On-board Real-time Pose Estimation for UAVs using Deformable Visual Contour Registration. In: Proceedings of the IEEE International Conference in Robotics and Automation (ICRA)

Arras KO, Mozos OM, Burgard W (2007) Using boosted features for the detection of people in 2D range data. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), pp 3402–3407

Arulampalam M, Maskell S, Gordon N, Clapp T (2002) A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. IEEE Transactions on Signal Processing 50(2):174–188

Blackman SS (2004) Multiple hypothesis tracking for multiple target tracking. IEEE Aerospace and Electronic Systems Magazine 19(1):5–18

Brscic D, Kanda T, Ikeda T, Miyashita T (2013) Person tracking in large public spaces using 3-d range sensors. IEEE Transactions on Human-Machine Systems 43(6):522 – 534

Burgard W, Moors M, Stachniss C, Schneider FE (2005) Coordinated multi-robot exploration. IEEE Transactions on Robotics 21(3):376–386

Capitan J, Merino L, Ollero A (2016) Cooperative decision-making under uncertainties for multi-target surveillance with multiples UAVs. Journal of Intelligent & Robotic Systems 84(1):371–386

Charrow B, Michael N, Kumar V (2013) Cooperative multi-robot estimation and control for radio source localization. In: Desai PJ, Dudek G, Khatib O, Kumar V (eds) Experimental Robotics: The 13th International Symposium on Experimental Robotics, Springer International Publishing, Heidelberg, pp 337–351

Choi W, Pantofaru C, Savarese S (2011) Detecting and tracking people using an RGB-D camera via multiple detector fusion. In: Workshop on Challenges and Opportunities in Robot Perception (in conjunction with ICCV-11)

Chung T, Hollinger G, Isler V (2011) Search and pursuit-evasion in mobile robotics. Autonomous Robots 31(4):299–316

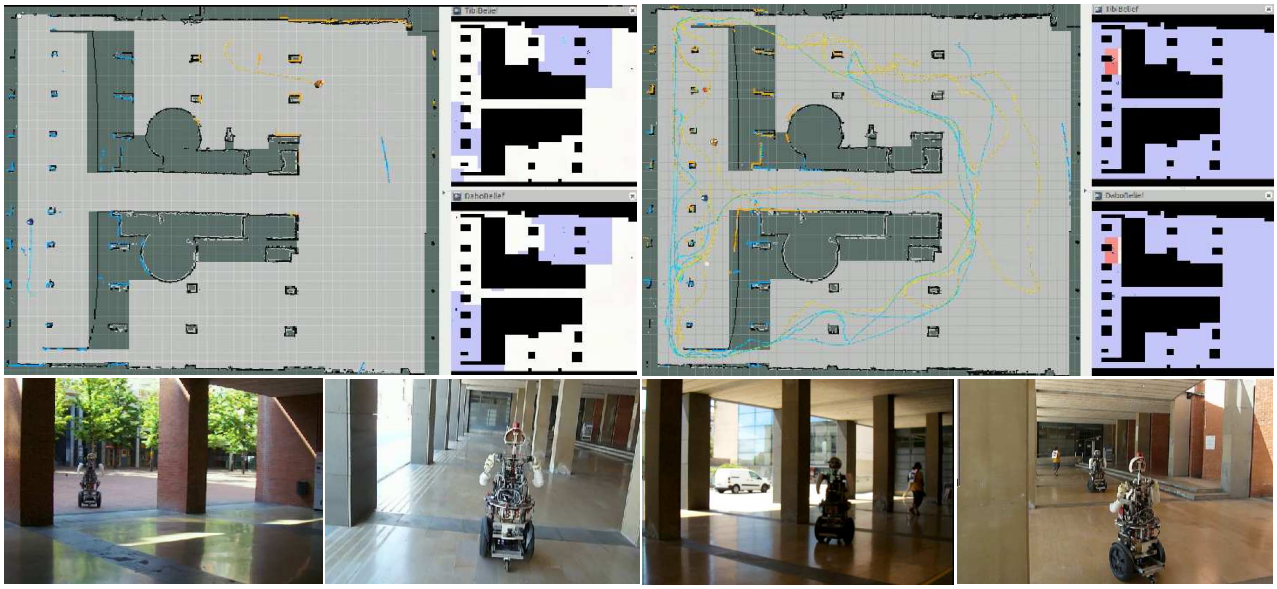

Fig. 7 Two different scenes during the experiments where the robots search for and track the person. The large maps show the robots (blue and orange) and the trajectories they have executed; the red circle indicates that the person has been detected at that location, and a white circle means that the person was detected there. The smaller maps represent the beliefs of Tibi (up) and Dabo (down): black squares are obstacles, the blue circles are the robots, and the white to red squares refer to a low to high probability of the person being there.

Cui J, Zha H, Zhao H, Shibasaki R (2008) Multi-modal tracking of people using laser scanners and video camera. Image and Vision Computing 26(2):240 – 252

Ferrein A, Steinbauer G (2016) 20 years of robocup. KI - Künstliche Intelligenz 30(3):225–232

Garrell A, Sanfeliu A (2012) Cooperative social robots to accompany groups of people. The International Journal of Robotics Research 31(13):1675–1701

Garrell A, Villamizar M, Moreno-Noguer F, Sanfeliu A (2013) Proactive behavior of an autonomous mobile robot for human-assisted learning. In: Proceedings of IEEE RO-MAN, pp 107–113

Glas DF, Morales Y, Kanda T, Ishiguro H, Hagita N (2015) Simultaneous people tracking and robot localization in dynamic social spaces. Autonomous Robots 39(1):43–63

Goldhoorn A, Alquézar R, Sanfeliu A (2013a) Analysis of methods for playing human robot hide-and-seek in a simple real world urban environment. In: ROBOT (2), Springer, Advances in Intelligent Systems and Computing, vol 253, pp 505–520

Goldhoorn A, Alquézar R, Sanfeliu A (2013b) Comparison of MOMDP and heuristic methods to play hide-and-seek. In: Gibert K, Botti VJ, Bolaño RR (eds) CCIA, IOS Press, Frontiers in Artificial Intelligence and Applications, vol 256, pp 31–40

Goldhoorn A, Garrell A, Alquézar R, Sanfeliu A (2014) Continuous real time pomcp to find-and-follow people by a humanoid service robot. In: Proceedings of the IEEE-RAS International Conference on Humanoid Robots, pp 741–747

Grisetti G, Stachniss C, Burgard W (2007) Improved techniques for grid mapping with rao-blackwellized particle filters. Journal IEEE Transactions on Robotics 23(1):34–46

Hlinka O, Hlawatsch F, Djuric PM (2013) Distributed particle filtering in agent networks: A survey, classification, and comparison. IEEE Signal Processing Magazine 30:61–81

Hollinger G, Yerramalli S, Singh S, Mitra U, Sukhatme G (2015) Distributed data fusion for multirobot search. IEEE Transactions on Robotics 31(1):55–66

Hollinger GA, Singh S, Kehagias A (2010) Improving the efficiency of clearing with multi-agent teams. International Journal of Robotics Research 29(8):1088–1105

Johansson E, Balkenius C (2005) It's a child's game: Investigating cognitive development with playing robots. In: Proceedings of the 4th International Conference on Development and Learning, pp 164–164

Kurniawati H, Hsu D, Lee W (2008) SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. In: Proceedings of Robotics: Science and Systems IV, Zurich, Switzerland

Lian FL, Chen CL, Chou CC (2015) Tracking and following algorithms for mobile robots for service activities in dynamic environments. International Journal of Automation and Smart Technology 5(1):49–60

Linder T, Breuers S, Leibe B, Arras KO (2016) On multi-modal people tracking from mobile platforms in very crowded and dynamic environments. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), pp 5512–5519

Luber M, Sinello L, Arras K (2011) People tracking in RGB-D data with on-line boosted target models. In: Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS), pp 3844–3849

Marconi L, Melchiorri C, Beetz M, Pangercic D, Siegwart R, Leutenegger S, Carloni R, Stramigioli S, Bruyninckx H, Doherty P, Kleiner A, Lippiello V, Finzi A, Siciliano B, Sala A, Tomatis N (2012) The SHERPA project: Smart collaboration between humans and ground-aerial robots for improving rescuing activities in alpine environments. In: Proceedings of the IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR), pp 1–4

Montemerlo M, Thrun S, Whittaker W (2002) Conditional particle filters for simultaneous mobile robot localization and people-tracking. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), IEEE, vol 1, pp 695–701

Ong SCW, Png SW, Hsu D, Lee WS (2010) Planning under Uncertainty for Robotic Tasks with Mixed Observability. International Journal of Robotics Research 29(8):1053–1068

Oyama T, Yoshida E, Kobayashi Y, Kuno Y (2013) Tracking visitors with sensor poles for robot's museum guide tour. In: Proceedings of the 6th International Conference on Human System Interactions

(HSI), IEEE, pp 645–650
Pineau J, Gordon G, Thrun S (2003) Point-based value iteration: An anytime algorithm for POMDPs. In: Proceedings of the International Joint Conference on Artificial Intelligence, pp 477–484
Sanfeliu A, Andrade-Cetto J, Barbosa M, Bowden R, Capitán J, Corominas A, Gilbert A, Illingworth J, Merino L, Mirats JM, Moreno P, Ollero A, Sequeira Ja, Spaan MTJ (2010) Decentralized Sensor Fusion for Ubiquitous Networking Robotics in Urban Areas. Sensors 10(3):2274–2314
Sheh R, Schwertfeger S, Visser A (2016) 16 years of robocup rescue. KI - Künstliche Intelligenz 30(3):267–277
Sheng X, Hu YH, Ramanathan P (2005) Distributed particle filter with GMM approximation for multiple targets localization and tracking in wireless sensor network. In: Proceedings of the 4th international symposium on Information processing in sensor networks (IPSN), IEEE Press, Piscataway, NJ, USA
Silver D, Veness J (2010) Monte-Carlo planning in large POMDPs. Proceedings of 24th Advances in Neural Information Processing Systems (NIPS) pp 1–9
Thrun S, Fox D, Burgard W, Dellaert F (2001) Robust Monte Carlo localization for mobile robots. Artificial Intelligence 128(1–2):99–141
Thrun S, Burgard W, Fox D (2005) Probabilistic Robotics (Intelligent Robotics and Autonomous Agents). The MIT Press
Volkhardt M, Gross HM (2013) Finding people in apartments with a mobile robot. In: IEEE International Conference on Systems, Man, and Cybernetics, pp 4348–4353
Vázquez MA, Míguez J (2017) A robust scheme for distributed particle filtering in wireless sensors networks. Signal Processing 131:190 – 201
Xu Z, Fitch R, Sukkarieh S (2013) Decentralised coordination of mobile robots for target tracking with learnt utility models. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), IEEE, pp 2014–2020