Continuous Real Time POMCP to *Find-and-Follow People* by a Humanoid Service Robot

Alex Goldhoorn, Anaís Garrell, René Alquézar and Alberto Sanfeliu

***Abstract*— This study describes and evaluates two new methods for finding and following people in urban settings using a humanoid service robot: the *Continuous Real-time POMCP* method, and its improved extension called *Adaptive Highest Belief Continuous Real-time POMCP follower*. They are able to run in real-time, in large continuous environments. These methods make use of the online search algorithm *Partially Observable Monte-Carlo Planning* (POMCP), which in contrast to other previous approaches, can plan under uncertainty on large state spaces. We compare our new methods with a heuristic person follower and demonstrate that they obtain better results by testing them extensively in both simulated and real-life experiments. More than two hours, over 3 km, of autonomous navigation during real-life experiments have been done with a mobile humanoid robot in urban environments.**

I. INTRODUCTION

The importance of autonomous mobile robots in industrial and research applications is growing, therefore, the need to execute successfully challenging missions and tasks has also grown. Thus, humanoid urban robots require several behaviors in order to successfully serve people. In this work, we are interested in how the robot can find and follow people in cluttered, dynamic environments, in order to help him or her.

Research into Human-Robot Interaction (HRI) in the field of *find-and-follow* is still new in comparison to traditional service robotics, such as HRI on a campus [1]. Therefore, prior research in this particular field is relatively minimal [2]. Most of the current research studies robots that participate in social human interactions as companions [3]. *Find-and-follow* is a multidisciplinary field of robotics in which a mixture of subjects such as perception, robot navigation and HRI intervenes. Despite the heterogeneity of the subjects treated, the problem can not be tackled independently but in a holistic way, which is not an easy endeavor. In this paper, we present two new methods: the *Continuous Real-time POMCP* (CR-POMCP) model, and its improved extension called *Adaptative Highest Belief Continuous Real-time POMCP Follower* (*Adaptive HB-CR-POMCP Follower*), which make use of POMCP (*Partially Observable Monte-Carlo Planning* [4]). The *find-and-follow* task is executed by the humanoid robot in an on-line real-time fashion, using continuous state space on large maps. In addition, the system takes into account sensory noise in the localization.

This work has been partially funded by the DPI2013-42458-P.

The authors are with the Institut de Robòtica i Informàtica Industrial (CSIC-UPC). Llorens Artigas 4-6, 08028 Barcelona, Spain. {agoldhoorn,agarrell,ralqueza,sanfeliu}@iri.upc.edu

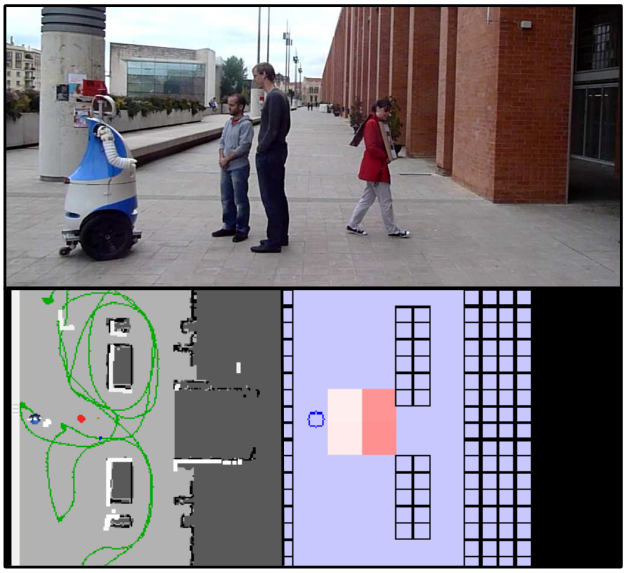

Fig. 1. *Top:* Dabo performs the find-and-follow task with a target. *Bottom:* Robot's estimation of target's position (belief).

The CR-POMCP extends the classical POMCP model, working in the continuous space instead of the discrete space, plans the action in real-time. In this way we can do real-time experiments with human volunteers. Moreover, we compare our methods with a previous and well-known system: heuristic person follower [5] Through a large number of synthetic and real experiments, we observed that the *Adaptive Highest Belief Continuous Real-time POMCP Follower* was able to find and follow different volunteers, while the other two methods performed the task slowly or lost the person, and therefore could not continue the mission.

Finally, the validation of the model is accomplished throughout an extensive set of simulations and real-life experiments. Our results total over 3 km of autonomous navigation, and more than two hours during over a week of experimentation.

In the reminder of the paper we start by introducing the Related Work. Section III provides the different methods we propose. Finally, the results of the methods and conclusions are presented in Sections IV and V, respectively.

II. RELATED WORK

Robotic research has the potential to help humans in urban services, here the robot is enabled to support humans by finding and accompanying them. Research into Human-Robot Interaction in the field of urban services is still relatively new in comparison to traditional research. In this

work, we present *CR-POMCP* (and the improved *Adaptive HB-CR-POMCP Follower*), where an autonomous robot is able to find a person in a dynamic cluttered environment and accompany him/her until a goal. For this reason, our topics of interest are: safe find-and-follow navigation with humans and accompanying people.

In the last several decades, major technological achievements have been accomplished with respect to the development of autonomous mobile robots for outdoor environments. Robust and reliable systems for navigation [6], obstacle avoidance [7], and localization [8] have been successfully integrated into several kinds of robotic platforms [9]. A number of methods have been developed to allow robots to navigate safely around people in specific, typically non-generalizable tasks [10].

Many works have been presented on finding people in indoors environments. In [11] a service robot is able to detect and find people inside a house making use of their legs, face, body-shape and motion. Nevertheless this work can not be applied to outdoor environments and it is assumed that the person is on the same position during the experiment, which is a strong assumption. Johanson et al. [12] presented a simplification of the real world problem of finding people, the hide-and-seek game, where there is a robot seeking and one or more players hiding. The hide-and-seek game also requires a high number of cognitive functions such as: search and navigation, finding coordination, anticipation and planning.

In [13] we focused on the hide-and-seek game, because this gives a limited and easier working environment to study the problem of searching people and to test the methods. First, we started to work in discrete time and space [14], and we used MOMDPs (Mixed Observable Markovian Decision Processes, [15]) to search for a person.

As opposed to previous works, in order to find a person our method only requires a map of the environment and is able to continuously plan its decisions based on people's behavior. Moreover, our method gives a probability map to indicate the possible positions of the person who is being searched for.

III. FIND-AND-FOLLOW METHODS

For a robot to find and follow a person we present three methods: 1) the *Heuristic Follower* which is used as comparison; 2) *Continuous Real-time Partially Observable Monte-Carlo Planning* (CR-POMCP) which is much smarter keeping in mind movement, and has memory; and 3) an extension to the previous, the *Adaptive Highest Belief CR-POMCP Follower*.

A. HEURISTIC FOLLOWER

The easiest way for the robot to follow a person is by directly going to the position where he or she is detected [16]. The robot's goal position is set just in front of the person. When the target is not visible anymore, the robot keeps going to the last person's position. When reaching this position, the robot stays and waits for the person to appear again.

B. CONTINUOUS REAL-TIME POMCP

Whereas the *Heuristic Follower* only uses the last person's position, here we present a method that takes into account the future movement of the robot and person, sensory noise, and maintains a memory of the possible positions of the person.

POMCP (*Partially Observable Monte-Carlo Planning* [4]) is a Reinforcement Learning algorithm for planning under uncertainty, which is present in our problem due to not always knowing the location of the person we are looking for and due to sensory noise.

POMCP is based on and makes use of the *Partially Observable Markovian Decision Process* (POMDP [17], [18]), as we will explain firstly. Thereafter, the POMCP algorithm will be explained first for the discrete case, then we present how to use it in continuous space states. Moreover, all the planning is done in *real-time*.

1) POMDP: A POMDP model contains a set of states ($\mathcal{S}$), which for our case are defined as the position of the robot and person: $(s_{\text{robot}}, s_{\text{person}})$. The robot can do an action out of the limited set $\mathcal{A}$. Here, the robot is able to move in one of eight directions, or stay at the same place. Instead of knowing the exact state, an observation of the state is done. In the find-and-follow problem observations ($\mathcal{O}$) are equal to the states, but the person's position (s_{person}) has a special value *hidden* when he or she is not visible. The probabilities of going from one state s to another s' with an action a are defined by $\mathcal{T} = P(s'|s, a)$, and the observation probability by $\mathcal{Z} = P(o|s', a)$. The reward function $\mathcal{R}$ is used to guide the learning process indicating which are the best actions to do in which states, the policy. Our reward function, $-d_{rp}$, is increasing when the robot-person distance d_{rp} is decreasing.

Instead of knowing the full state, a probability of being in each possible state is stored, this is called the *belief*. The starting belief b_0 has to be given in advance, for example a uniformly distributed probability over all the locations where the person might be hidden. Thereafter, the belief is updated using the observation and the probability functions. In Fig. 1-*bottom-right* the belief location is represented by a high probability (red) to a low probability (white). The resolution of the belief is low because the map size is big, and it keeps in mind possible sensory noise.

The best action to execute for each (belief) state is calculated by computing the value function [19], [20], [18]:

$$Q(b, a) = R(b, a) + \gamma \sum_{o \in O} P(o|b, a) V(\tau(b, a, o)) \qquad (1)$$

where $R(b, a)$ is the reward for the belief state b and action a, γ is the discount factor, $V(b) = \max_{a \in A} Q(b, a)$, and $\tau(b, a, o)$ is the next belief state b'. Finding the exact policy would be intractable (PSPACE-hard, [21]), therefore approximation methods are used which sample the belief space in a smart way to find a policy [18], [17]. In general POMDPs have two main problems [15]: 1) the growth of the belief space (*curse of dimensionality*), and 2) the exponential tree growth with the search horizon (*curse of history*).

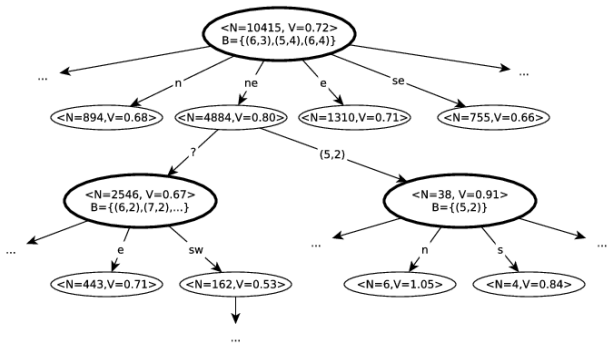

Fig. 2. A part of a policy tree generated by POMCP during a find-and-follow task. The root represents the current situation and contains a belief. For each action (here representing a movement: **n**orth, **n**ort**h**east, **s**outh, etc.) a child node is reached. From the action nodes, an observation ('?' indicating that the person is not visible) can be taken which reaches a new situation. The root and observation nodes of the first layer contain a belief. All nodes maintain an expected value V and the number of times N this value has been updated.

In previous work [13], [14] we tried to tackle this problem by using a layered MOMDP, but this was achieved for relatively small maps. Other works [15], [22], [17], [18] showed simulations of hide-and-seek like problems using POMDPs, but in small discrete environments, and only few real-life experiments [18], [14] were done.

2) POMCP: The use of POMDP models is limited due to computational complexity and exponential memory growth. An improvement of *Monte Carlo Value Iteration* (MCVI [23]) is to calculate the expected reward over a random set of samples instead over all states.

The idea of using Monte Carlo simulations is extended by Silver and Veness [4] by doing Monte Carlo simulations to generate a policy. The big advantage of POMCPs is that they tackle the *curse of history* and the *curse of dimensionality* by simulating the POMDP, and not requiring the whole fully defined model. Thus, the complexity depends on the POMDP simulator. Convergence for POMCP with finite horizon is proven in [4], but can be extended to the infinite case as shown in [24].

3) ALGORITHM: POMCP generates a policy tree with two types of nodes: *belief nodes* that represent a belief state, and *action nodes* which are their direct children and are reached by doing an action, see Fig. 2. The root is a *belief node* and contains the belief, which in POMCPs is represented by a list of states instead of probabilities of all possible discrete states. When a state is more probable then the state is repeated more times in this list.

Whereas the POMDP solvers mainly use value iteration (1) to find a policy, POMCP makes use of Monte Carlo simulations. Each node in the tree keeps track of the average expected reward V, and the number of times N a simulation passes through the node. Before the first learning iteration is executed, an initial belief has to be set for the root's belief *Root*.$\mathcal{B}$. The POMDP simulator's function $\mathcal{I}(o_0)$ is used to generate n_{belief} states of the initial belief (explained in detail later on in subsection III-B.5).

Algorithm 1 The POMCP planner. Retrieving children nodes is noted as *Node*[a] (for action a for example).

```
1: function SIMNODE(Node,s,depth)
2:     if depth > d_max then return 0
3:     else
4:         a ← argmax_a Node[a].V + c sqrt(log(Node.N) / Node[a].N)
5:         if depth = 1 then Node.B = Node.B ∪ {s}
6:         (s', o, r_immediate) ← G(s, a)
7:         if s' is not final and not Node[a][o] exists and
8:                                 Node[a].N ≥ e_count then
9:             Add Node[a][o]
10:        end if
11:        if s' is not final then
12:            if Node[a][o] exists then
13:                r_delayed ← SIMNODE(Node[a][o],s',depth+1)
14:            else
15:                r_delayed ← ROLLOUT(s',depth+1)
16:            end if
17:        else
18:            r_delayed ← 0
19:        end if
20:        r_total ← r_immediate + γ r_delayed
21:        Node[a].N ← Node[a].N + 1
22:        Node[a].V ← Node[a].V + (r_total − Node[a].V) / Node[a].N
23:        Node.N ← Node.N + 1
24:        Node.V ← Node.V + (r_total − Node.V) / Node.N
25:        return r_total
26:    end if
27: end function
28: function ROLLOUT(s,depth)
29:    if depth > d_max then return 0
30:    else
31:        a ~ π_rollout()
32:        (s', o, r) ← G(s, a)
33:        return r + γ ROLLOUT(s',depth+1)
34:    end if
35: end function
```

Before each robot's step the policy tree is updated by doing n_{sim} simulations. Randomly a state $s \sim$ *Root*.$\mathcal{B}$ is sampled, and then SIMNODE(*Root*,s,0) in Algorithm 1 is called. An overview of the parameters used in POMCP is given in Table I. Firstly, an action is chosen in line 4 based on the highest value V of the action node and an exploration factor [4], which is weighted by the exploration constant c. This latest factor causes the learning process to explore other possible actions instead of only exploiting the current knowledge.

The POMDP simulator $\mathcal{G}$ in line 6 returns the new state s', observation o, and reward $r_{\text{immediate}}$ based on the current state s and chosen action a. The tree is traversed by choosing the edges for a and o respectively; if this node does not yet exist, s' is not a final state and the simulation already passed at least e_{count} times before, then a new belief node is added (line 9). The expand count (e_{count}) prevents the tree from growing too fast.

Then, if the state is not final, the future reward r_{delayed} is calculated by using the child node, if it exists in line 13, or by executing the ROLLOUT function (line 28). A rollout policy π_{rollout} [4] is followed until the run has finished or the maximum depth has been reached. This policy can be random, as we used, or it can be based on a heuristic.

TABLE I

THE PARAMETERS OF THE POMCP ALGORITHM.

Parameter	Description
γ	Discount factor.
n_{sim}	Number of Monte Carlo simulations.
n_{belief}	Number of states in the belief.
c	Exploration constant.
e_{count}	Expand count.
d_{max}	Maximum tree depth.

In line 20 the reward for the current belief and action node r_{total} are calculated, where r_{delayed} is weighted with the discount factor γ to reduce the influence of possible future rewards. Next the average values and new counts of the nodes are updated (lines 21-24).

Finally, after the policy tree has been learned, the action to execute can be chosen from the tree: $a \leftarrow \text{argmax}_a \textit{Root}[\text{a}].V$.

After having executed an action a, a new observation o is done. The new belief node can be retrieved from the tree by following edge a, which reaches an action node, and with o reaching the new belief node. If for example, in Fig. 2, action *ne* has been taken, and observation *hidden* ('?') has been done then this would reach the new big node left-under. This new node is then taken as root, and if this child *belief node* does not exist, a new *belief node* is created. In simulation the beliefs of the belief nodes at depth 1 already have been grown (line 5 in Algorithm 1), and when this node has been chosen as new root, implies that it already will have some states in the belief.

A minimum number of n_{belief} states in the belief are maintained to assure a minimal spread of the belief over the possible person's locations. If the new root's belief includes less than n_{belief} states, then new states are generated for the belief using the POMDP simulator: $(s', o_{\text{sim}}, r) = \mathcal{G}(s, a)$, where s is sampled from the previous root's belief. The state s' is stored in the new root's belief if it matches the real observation $o = o_{\text{sim}}$. If there is no belief in the new root, an initial belief can be calculated from the new observation, but this results in losing the knowledge about the state (belief).

4) POMCP IN CONTINUOUS SPACE (CR-POMCP): The use of continuous state space in POMDPs is not evident, furthermore the POMDP's belief space is continuous with an infinite number of dimensions. However Porta et al. [25] show that value iteration can be used due to properties of the value function, but the policy's complexity grows. *Monte Carlo Value Iteration* (MCVI) is used in [26] to find a policy for POMDPs. A policy graph is generated to avoid having to handle the continuous belief space. A number of N states are randomly sampled to estimate the expected reward.

In contrast to the previously mentioned methods, we do not have to define a POMDP model, but we simulate the transitions. This avoids us from having to handle belief spaces of infinite dimensions.

Using continuous states in POMCPs requires us to 1) define the continuous states; 2) create a POMDP simulator that handles these states; and 3) assure that the observations are discretized. In continuous state space an exact position is used instead of grid cells. The actions and observations however always have to be discrete, otherwise the planner would have to take into account too many situations, since the policy tree could grow too wide and the values in the nodes would not converge.

A balance of the observation discretization is important, because very small grid cells imply a higher resolution, but wide policy trees with lower probability of convergence to a good policy. In contrary lower detail can cause problems of precision in the planning, but increases probability of convergence.

5) FIND-AND-FOLLOW POMDP SIMULATOR: Up to this section we have explained how the CR-POMCP works, now we will explain the details of the POMDP simulator $\mathcal{G}$ (line 6 in Algorithm 1) as defined for the find-and-follow task. The map is known beforehand, which is discrete with each grid cell being either an *obstacle* or *free*. States and observations are the position of the seeker and hider $(s_{\text{robot}}, s_{\text{person}})$, and person's observation o_{person} can be *hidden*. There are nine actions: eight movements (north, northeast, etc.) and standing still.

Initial states are generated with the function $s = \mathcal{I}(o_0)$, where o_0 is the initial observation. The generated state is equal to the observation, but when the person's observed position is *hidden* then s_{person} is chosen randomly from hidden positions as seen from the location o_{robot}.

New states are generated by $(s', o, r) = \mathcal{G}(s, a)$, based on the current state and an action. The new state variable s'_{robot} depends on the action and robot's position s_{robot}; the person's movement is modeled as random, but could be modeled heuristically. The observation o equals the new state s', but o_{person} =*hidden* when the person is not visible according to a ray trace algorithm. The reward function is heuristic: $r = -d_{rp}$, where d_{rp} is the shortest path distance between the robot and person.

Gaussian noise is added to the state and observation: $\mathcal{N}(0, \sigma)$ in order to simulate sensory noise. Different standard deviations of the noise are defined for the next seeker state (σ_{ns}), next person state (σ_{np}), seeker observation (σ_{os}), and person observation (σ_{op}). False negatives are simulated by converting a person's observation o_{person} in *hidden* with probability p_{fn}, and false positives by generating a random position o_{person} with probability p_{fp} when the person is not visible.

C. ADAPTIVE HB-CR-POMCP FOLLOWER

When executing the find-and-follow task using the *CR-POMCP* method, our real robot Dabo [9] moved too slowly, not being able to follow the person in real-time. This was caused by the slightly deviating discrete actions due to the random Monte-Carlo simulations; for example first an action north, and then the action northeast.

To solve this problem, farther goal positions have to be given to the robot, such that it uses its navigation system to reach the goal without unnecessary turns. As goal the point with the highest belief is passed to the robot. The goal is updated every $t_{\text{hb.update}}$ s to prevent the goal from being changed too often.

Fig. 3. *Left: Barcelona Robot Lab, North campus of the UPC. Right: FME Lab, South Campus of the UPC.*

This method, the *Adaptive Highest Belief CR-POMCP Follower*, makes use of the previously discussed method, *CR-POMCP*, to update the belief in each iteration. When the person is visible, the *Heuristic Follower* is used to follow the person, but at the same time the belief of the person's position is updated. When the person is not visible, a 2D histogram from the belief of the position of the person (s_{person}) is generated. From this matrix the cell with the highest count is used as the robot's next goal.

It is important to choose the right resolution for the matrix, because a high resolution results in a high number of cells which requires a higher number of belief points (n_{belief}), and therefore more simulations n_{sim}. A too low resolution reduces the precision with which the robot can be sent to find the person.

D. SUMMARY

To sum up, the *Heuristic Follower* simply goes to the last visible position of the person. A smarter method is the *CR-POMCP* which is able to find and follow a possibly hidden person. It takes into account movement of the robot and person, and makes real-time usage of POMCP in a continuous state space. Finally, an extension on the latter is presented, the *Adaptive HB-CR-POMCP Follower* which improves the movement behavior of the mobile robot.

IV. SIMULATIONS AND EXPERIMENTS

In this section we introduce the results obtained from the simulations and the real-life experimentation.

A. EXPERIMENTAL SETTING

Here, the robotic platform and the testing environments used in the present paper will be presented.

1) ROBOT PLATFORM: To conduct all the experiments and to test the approach presented, we used two twin mobile service robots, called Tibi and Dabo, each designed to work in urban pedestrian areas and to interact with people, see Fig. 3. They are based on a two-wheeled Segway RMP200 platform, which works as an inverted pendulum in constant balancing. They can rotate on the spot (nonholonomic), and have wheel encoders providing odometry, and inclinometers providing pitch and roll data.

As social robots, Tibi and Dabo are designed to interact with people. They are equipped with several interaction devices to enable them to engage with friendly interactions, such as a touchable screen, speaker, movable arms and head, and LED illuminated facial expressions.

2) ENVIRONMENTS: The experiments were conducted at the BRL (Barcelona Robot Lab) and the FME (Facultat de Matemàtiques i Estadística) lab, outdoor urban environments located at the North and South Campus of the Universitat Politècnica de Catalunya (UPC), respectively.

The BRL (Fig. 3-*Left*) is a large area of the campus that was outfitted as an experimental area, covering over $10.000\ m^2$, comprising six buildings and a central square, with multiple ramps, staircases, and typical obstacles such as bulletin boards, bicycle stands, trashcans and flower pots. The area used for our experimens was discretized to 80×15 grid cells, 1 m each. The FME lab (Fig. 3-*right*) consists of a green space and a paved area, separated by stairs. In this area the grid cells where of 1 m, having in total 17×12 cells.

B. SIMULATIONS

The experiments are done on maps with discrete grid cells which either can be *free* or an *obstacle*. For each iteration step the robot and person can move in one of eight discrete directions (north, northeast, south, etc.), or stay at the same position. The coordinates are continuous, and the movement distance is 1 cell per step for both agents (also in diagonal, i.e. not $\sqrt{2}$).

For simplicity the simulations do not include neither acceleration, nor friction, nor collision. The agents are not allowed to be neither outside the map nor on top of an obstacle. For algorithms that give goals farther away, such as the *Adaptive HB-CR-POMCP Follower*, the shortest path is calculated, and during each iteration one step is taken.

The person simulation is done by starting at a random position, and by giving it a random goal. Each iteration the goal is approached one step, and when the goal is reached, a new random goal is chosen. A multi-person environment was simulated by adding a group of 10 persons which cause occlusions, and have the same behavior as the person.

The three algorithms discussed in Section III were applied in simulations: 1) *Heuristic Follower*; 2) *CR-POMCP*; 3) *Adaptive HB-CR-POMCP Follower*. The algorithms have been compared by looking at the average distance to the person.

To make the comparison between the three methods as fair as possible, for each run of simulations the robot's start position and the person's movement were the same. A total of more than 4000 simulations have been done repeating each of the conditions at least 40 times. For each simulation 200 steps were done, except for the smaller map FME, for which 100 steps were done.

The parameters of the algorithms which have been used in the simulations and real experiments were obtained experimentally, and are shown in Table II. In simulation the smaller values of n_{sim} and n_{belief} have been used for the small map only. The *Adaptive HB-CR-POMCP Follower* updates its belief every 3 s in the real experiments and every 3 iterations in the simulations. The 2D histogram used by the adaptive method has a resolution two times lower than the larger BRL Lab's grid map.

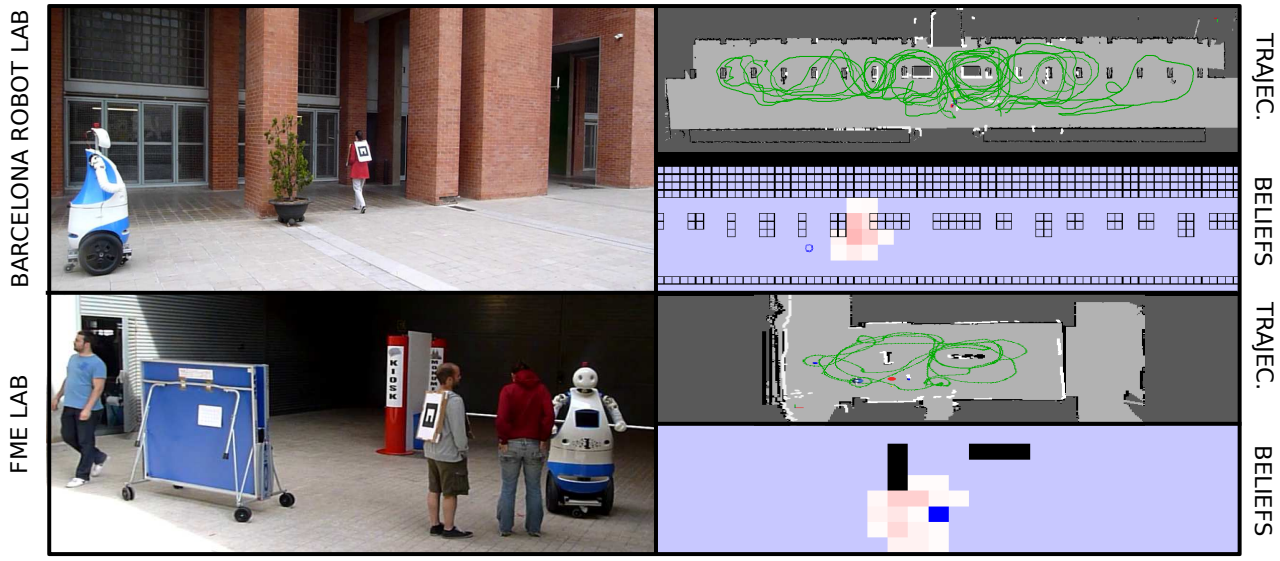

Fig. 4. Real-life experiments. *Left:* Some video captures of real experiments. Dabo performs the *find-and-follow* task. *Right:* The trajectory performed by the robot, and the same scene showing the belief of the person's location.

TABLE II

THE PARAMETERS VALUES USED DURING THE REAL AND SIMULATED EXPERIMENTS.

Parameter	Real	Sim.	Parameter	Real	Sim.
γ	0.95	0.95	σ_{np}	0.3	0.3
n_{sim}	1000	2500	σ_{os}	0.1	0.1
n_{belief}	500	1000	σ_{op}	0.1	0.1
c	$rows \times cols$		p_{fn}	0.3	0.3
e_{count}	2	2	p_{fp}	0	0.001
d_{max}	$2(rows \times cols)$		p_{rh}	0.01	0.01
σ_{ns}	0.2	0.2	$t_{\text{hb.update}}$	3 s	3 steps

TABLE III

		METHOD	Dist. [m]
FME LAB	0 People	Adaptive HB-CR-POMCP Follower	1.52 ($\pm$0.24)
		Heuristic Follower	1.70 ($\pm$0.31)
		Continuous Real-Time POMCP	2.33 ($\pm$0.23)
	10 People	Adaptive HB-CR-POMCP Follower	2.54 ($\pm$0.36)
		Heuristic Follower	3.11 ($\pm$0.47)
		Continuous Real-Time POMCP	3.26 ($\pm$0.36)
BRL LAB	0 People	Adaptive HB-CR-POMCP Follower	3.91 ($\pm$1.27)
		Heuristic Follower	6.36 ($\pm$1.71)
		Continuous Real-Time POMCP	5.59 ($\pm$1.26)
	10 People	Adaptive HB-CR-POMCP Follower	5.51 ($\pm$1.54)
		Heuristic Follower	7.25 ($\pm$1.68)
		Continuous Real-Time POMCP	6.81 ($\pm$1.58)

In Table III we show the average distances between the robot and the targets in our simulations. It can be seen, that in different environments and conditions the *Adaptive HB-CR-POMCP Follower* works much better than the other two approaches.

Finally, in order to evaluate if significant difference exists between the presented three methods, we used the Wilcoxon ranksum test, 2-sided. We can conclude that *Adaptive HB-CR-POMCP Follower* works better than the *Heuristic Follower* ($p < 0.01$) and *CR-POMCP* ($p < 0.01$) with or without other pedestrians. Comparing the *CR-POMCP* and the *Heuristic Follower*, we found that *CR-POMCP* is better ($p < 0.01$) for the large map BRL. For the smaller map FME, the *Heurstic Follower* is better than *CR-POMCP* ($p < 0.01$). The latter can be explained because the FME is a relatively small environment, and therefore the robot has a higher probability of re-detecting the person again. Moreover the *Heuristic Follower*'s movement is more efficient because it uses a shortest path planner, whereas *CR-POMCP* generates actions which are discrete and influenced lightly by the randomness of the Monte-Carlo simulations.

C. REAL-LIFE EXPERIMENTS

To evaluate the methods, more than one week of experiments were done with our mobile robot Dabo [9] in the two environments shown in Fig. 4. A total trajectory of more than 3 km was covered during all the successfully executed experiments, summing up to more than 3 hours. The lines in Fig. 4-*right* indicate the total trajectory executed by the robot.

In a single experiment, volunteers were used as person to follow, and others as dynamic obstacles, obstructing the robot's vision by passing in front of the robot or by standing in a group of people. The person to be followed was told to start at a location not too far, hidden or not to the robot. The robot did not move quickly for security reasons, and therefore the volunteer was also asked not to move too fast. From then on the person could walk, and let the robot find and follow him/her. The person wore a tag, see Fig. 4-*left*, such that the robot was able to recognize him or her.

In order to validate the models in real experiments, we have compared our approach with respect to the *Heuristic Follower*. Experiments in the FME lab showed that the robot using the *Heuristic Follower* did not move most of the time (50.4%; $p < 0.001$, Fisher's exact test), therefore, the robot was not able to follow the target. In contrast, using the *Adaptive HB-CR-POMCP Follower* the robot moved all the time when the person was not visible, except for 3.7% of the time, and therefore it could follow the target. The *CR-POMCP* method alone showed promising results

in simulation, but in the real-life experiments it moved too slowly to follow the person.

Extensive experiments with the *Adaptive HB-CR-POMCP Follower* showed that for the BRL map, over all experiments, the person was visible 57% of the time, and the average distance to the person when visible was 2.7 ± 1.8 m (average $\pm$ standard deviation). For the FME environment 48% and 3.6 ± 1.8 m respectively.

For further information, check the videos of the experimental results in the project web `http://www.iri.upc.edu/groups/lrobots/find-and-follow/humanoids2014.html`.

V. CONCLUSIONS

This work has presented a find-and-follow behavior for a humanoid service robot, in order to serve humans in an urban environment and thereby improving Human-Robot Interaction. The presented methods in real-life experiments work on-line, real time in a large continuous space with noisy sensor information.

The models have been tested in simulations which showed us the robustness to the occlusions of dynamic obstacles, i.e. other pedestrians walking in the same environment. The *CR-POMCP* method did not work significantly better than the *Heuristic Follower* in the smaller map, due to the slow motion of our robot, produced by the discrete actions. The improved method *Adaptive HB-CR-POMCP Follower* worked significantly better than the previous ones, by combining the smart *CR-POMCP* with the *Heuristic Follower*.

Real-life experiments, where the robot covered several kilometers, were done to test our methods. The *Adaptive HB-CR-POMCP Follower* showed that in all urban environments, the robot was able to follow the target and find him/her in all the experiments performed with volunteers.

Next we would like to use a smarter prediction of the person's path, instead of a random movement [5]. Furthermore the adaptive method could be improved by giving preference to close search locations.

REFERENCES

[1] A. Garrell, M. Villamizar, F. Moreno-Noguer, and A. Sanfeliu, "Proactive behavior of an autonomous mobile robot for human-assisted learning," in *IEEE RO-MAN*. IEEE, 2013, pp. 107–113.

[2] J. Kiener and O. Von Stryk, "Towards cooperation of heterogeneous, autonomous robots: A case study of humanoid and wheeled robots," *Robotics and Autonomous Systems*, vol. 58, no. 7, pp. 921–929, 2010.

[3] B. Robins, E. Ferrari, K. Dautenhahn, G. Kronreif, B. Prazak-Aram, G.-j. Gelderblom, B. Tanja, F. Caprino, E. Laudanna, and P. Marti, "Human-centred design methods: Developing scenarios for robot assisted play informed by user panels and field trials," *International Journal of Human-Computer Studies*, vol. 68, no. 12, pp. 873–898, 2010.

[4] D. Silver and J. Veness, "Monte-Carlo planning in large POMDPs," *24th Advances in Neural Information Processing Systems (NIPS)*, pp. 1–9, 2010.

[5] A. Garrell and A. Sanfeliu, "Local optimization of cooperative robot movements for guiding and regrouping people in a guiding mission," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*. IEEE, 2010, pp. 3294–3299.

[6] G. Lidoris, F. Rohrmuller, D. Wollherr, and M. Buss, "The autonomous city explorer (ace) project-mobile robot navigation in highly populated urban environments," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2009, pp. 1416–1422.

[7] V. Desaraju, H. Ro, M. Yang, E. Tay, S. Roth, and D. Del Vecchio, "Partial order techniques for vehicle collision avoidance: Application to an autonomous roundabout test-bed," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2009, pp. 82–87.

[8] A. Corominas-Murtra, J. Mirats-Tur, and A. Sanfeliu, "Efficient active global localization for mobile robots operating in large and cooperative environments," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2008, pp. 2758–2763.

[9] A. Garrell and A. Sanfeliu, "Cooperative social robots to accompany groups of people," *The International Journal of Robotics Research*, vol. 31, no. 13, pp. 1675–1701, 2012.

[10] M. Zinn, O. Khatib, B. Roth, and J. K. Salisbury, "Playing it safe [human-friendly robots]," *IEEE Robotics & Automation Magazine*, vol. 11, no. 2, pp. 12–21, 2004.

[11] M. Volkhardt, S. Müller, C. Schröter, and H.-M. Gross, "Playing hide and seek with a mobile companion robot," in *Humanoids*. IEEE, 2011, pp. 40–46.

[12] E. Johansson and C. Balkenius, "It's a child's game: Investigating cognitive development with playing robots," in *International Conference on Development and Learning*, 2005, p. 0:164.

[13] A. Goldhoorn, A. Sanfeliu, and R. Alquézar, "Comparison of momdp and heuristic methods to play hide-and-seek," in *CCIA*, ser. Frontiers in Artificial Intelligence and Applications, K. Gibert, V. J. Botti, and R. R. Bolaño, Eds., vol. 256. IOS Press, 2013, pp. 31–40.

[14] A. Goldhoorn, A. Sanfeliu, and R. Alquézar, "Analysis of methods for playing human robot hide-and-seek in a simple real world urban environment," in *ROBOT (2)*, ser. Advances in Intelligent Systems and Computing, M. A. Armada, A. Sanfeliu, and M. Ferre, Eds., vol. 253. Springer, 2013, pp. 505–520.

[15] S. C. W. Ong, S. W. Png, D. Hsu, and W. S. Lee, "Planning under Uncertainty for Robotic Tasks with Mixed Observability," *The International Journal of Robotics Research*, vol. 29, no. 8, pp. 1053–1068, May 2010.

[16] A. Garrell Zulueta, A. Corominas Murtra, and A. Sanfeliu Cortés, "Robot companions for guiding people in urban areas," in *ROBOT 2011 (III Workshop de Robótica)*, nov 2011.

[17] H. Kurniawati, D. Hsu, and W. Lee, "Sarsop: efficient point-based pomdp planning by approximating optimally reachable belief spaces," in *Robotics: Science and Systems, 2008*, 2008.

[18] J. Pineau, G. Gordon, and S. Thrun, "Point-based value iteration: An anytime algorithm for pomdps," in *International Joint Conference on Artificial Intelligence, 2003*, 2003, pp. 477–484.

[19] R. Bellman, "A markovian decision process," *Indiana Univ. Math. J.*, vol. 6, pp. 679–684, 1957.

[20] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.

[21] C. Papadimitriou and J. Tsisiklis, "The complexity of markov decision processes," *Mathematics of Operations Research*, vol. 12, no. 3, pp. 441–450, 1987.

[22] M. Araya-López, V. Thomas, O. Buffet, and F. Charpillet, "A closer look at MOMDPs," in *22nd International Conference on Tools with Artificial Intelligence - ICTAI*, 2010.

[23] Z. Lim, D. Hsu, and W. Lee, "Monte Carlo Value Iteration with Macro-Actions," in *25th Advances in Neural Information Processing Systems (NIPS)*, 2011, pp. 1287–1295.

[24] L. Kocsis and C. Szepesvári, "Bandit based monte-carlo planning," in *Proceedings of the 17th European Conference on Machine Learning*, ser. ECML'06. Berlin, Heidelberg: Springer-Verlag, 2006, pp. 282–293.

[25] J. M. Porta, M. T. J. Spaan, and N. Vlassis, "Robot planning in partially observable continuous domains," in *Robotics: Science and Systems*. MIT Press, 2005, pp. 217–224.

[26] H. Bai, D. Hsu, W. Lee, and V. Ngo, "Monte Carlo value iteration for continuous-state POMDPs," *Algorithmic Foundations of Robotics IX*, pp. 175–191, 2011.