Robotics and Autonomous Systems 98 (2017) 147–157

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

Searching and tracking people in urban environments with static and dynamic obstacles

Alex Goldhoorn *, Anaís Garrell, René Alquézar, Alberto Sanfeliu

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

HIGHLIGHTS

- We would like to emphasize that this work presents several advances in the state of the art.
- First, the modification on the Particle Filter (PF) to take into account when a person is lost in the tracking process, which implies that the observation is lost and the standard PF update weight mechanism cannot be used.
- We have designed a new updating mechanism of the weights of the Particle Filter (in our case the belief) which handles the lack of the observation, which is combined with the highest-belief method to decide the new goal of the robot (the highest-belief gives the most probable position of the person).
- Second, our model handles the search-and-track of a person in real-life experiments in a large campus map (60 m × 55 m), in continuous state space.
- The person can be searched and tracked even though he/she is occluded by static (environment obstacles) or by dynamic obstacles (other people moving).
- Moreover, the model allows person searching and tracking recovering when there is noise, false negative and positive detections and other perception problems due to the limitations of the sensors or the environment conditions.
- Finally, we contribute the new architecture described in Fig. 2, which handles all these situations.

ARTICLE INFO

Article history:
Available online 30 June 2017

Keywords:
Urban robotics
Search and rescue robots
Human–robot-interaction
Particle filter

ABSTRACT

Searching and tracking people in crowded urban areas where they can be occluded by static or dynamic obstacles is an important behavior for social robots which assist humans in urban outdoor environments. In this work, we propose a method that can handle in real-time searching and tracking people using a *Highest Belief Particle Filter Searcher and Tracker*. It makes use of a modified Particle Filter (PF), which, in contrast to other methods, can do both searching and tracking of a person under uncertainty, with false negative detections, lack of a person detection, in continuous space and real-time. Moreover, this method uses dynamic obstacles to improve the predicted possible location of the person. Comparisons have been made with our previous method, the *Adaptive Highest Belief Continuous Real-time POMCP Follower*, in different conditions and with dynamic obstacles. Real-life experiments have been done during two weeks with a mobile service robot in two urban environments of Barcelona with other people walking around.

1. Introduction

Mobile service robots should be able to search and track persons in urban environments in order to assist and serve people. However, these areas contain obstructions, such as static obstacles (e.g. walls, pillars, trees, etc.), and dynamic obstacles (e.g. other people walking around, passing bikes, cars, etc.) which make the task of searching and tracking a person difficult. And the detection failures and noise introduced by the sensors make it even more complex.

Research into human–robot interaction in the field of *search-and-track* is still new in comparison to traditional service robotics tasks, such as serving food in hospitals. Therefore, prior research in this particular field is relatively minimal. Most of the current research predominantly studies robots that participate in human–robot interaction, such as companions [1].

Another important application is the search-and-rescue task in urban environments where robots: (i) find and rescue victims in the rubble or debris as efficiently and safely as possible, and (ii) ensure that human rescue workers' lives are not put at great risk [2]. Generally, USAR (Urban Search and Rescue) environments are highly cluttered, and all robots that operate in these environments do not have a priori information about dynamical

* Corresponding author.
E-mail address: agoldhoorn@iri.upc.edu (A. Goldhoorn).

http://dx.doi.org/10.1016/j.robot.2017.06.005
0921-8890/

Fig. 1. Dabo performs the search-and-track task with a target (wearing a tag for recognition) in different urban environments.

obstacles in the scene. These conditions make it extremely difficult for robots to autonomously navigate in the scenes and identify victims, therefore, current applications of mobile robots in USAR operations require a human operator. Furthermore, autonomous urban service robots should be able to search and track people in a safe and natural way [3].

In this paper, we present a new method, the *Highest Belief Particle Filter Searcher and Tracker (HB-PF Searcher & Tracker)* for searching and tracking a person. Our presented method is able to work in urban environments with dynamic and static obstacles, under uncertainty, person's false detections, lack of a person's detection, in continuous space and in real-time. Furthermore, we present an extension which takes dynamic obstacles into account when predicting the person's location. Moreover, we compare our method with our previously presented *Adaptive Highest Belief CR-POMCP Follower* [4] in different areas with dynamic obstacles present, see Section 6.1. We do not require high precision for the location detection (i.e. within a few centimeters) since the robot should be able to interact with the person when knowing a rough direction and distance to the person. Our focus is not on precise localization, but on searching and tracking the person.

Additional considerations are required to make the system work properly. For example, sensory noise, normally Gaussian, is inevitable in real-life situations, and false negative and false positive detections tend to occur. The presented method takes the first two problems into account, and can handle situations with false positive detections of a short duration.

Finally, the validation of the method is accomplished throughout an extensive set of simulations and real-life experiments, see Fig. 1. For the later we accomplished of about 1.3 km of autonomous navigation, with more than an hour of successful experiments during two days of experimentation.

First, the related work is discussed, then Section 3 shows the system's architecture, Section 4 explains all the presented methods in detail. In Section 5, the experimental setup is introduced and in Section 6 the simulations and the real experiments are discussed. Finally, the last section contains the discussion and conclusions of our work.

2. Related work

In this work, we present several methods that can be used by an autonomous robot, in order to search and track a person in an urban dynamic cluttered environment.

Volkhardt and Gross [5] proposed a method to search and locate a person, thereby using a computationally expensive detection method, and an occurrence map—which indicates the most probable positions of the person on the map. Their search method was guided by a list of predefined navigation points, which were chosen based on the closeness and probability of occurrence of the person; the points were marked as visited if the person was not found. They assumed there to be one person, and if there were more, the robot went to the firstly detected person. The verification, in case of a false positive detection, was done by waiting for the person to interact with the robot. They did experiments with a real robot in a scenario with three rooms where the robot was able to locate the person up to 91% of the time.

Hollinger et al. [6] used multiple agents to search for a moving object in a known environment. They used graphs as maps, where the agent is assumed to have full visibility in each vertex. Besides simulations with multiple agents, they showed experiments with a real robot clearing an environment in cooperation with two people. In [7] they focused on data fusion between the agents, and they kept track of the probability of the person being in each of the vertices. Their experimentation was simulated, but they used real world maps.

Trafton et al. [8] used a cognitive architecture, ACT-R [9], to play hide-and-seek. Their research was based on experiments with a 3.5 year old child. They used a layered architecture in which the lower layer contained navigation, and the top layer the cognitive architecture. The created ACT-R module learned how to play hide-and-seek generating new rules. In [10], a human and robot were following an other person cooperatively. Both methods focused on cognitive methods, which require a high amount of symbolic knowledge of the world.

Many works that treat guiding and following do not handle situations in which the person to be followed is hidden, at most partly occluded, such as in [11]. In [12], the robot followed a person in order to navigate through crowded environments. Lian et al. [13] 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, and then a look-ahead algorithm (DWA*) to follow the target and avoid obstacles at the same time. A multi–people tracker using multiple cameras was presented in [14]. They used a Probabilistic Occupancy Map in which they can track several persons with high precision, allowing them to be partly occluded. In [15], a model that represents human activity events was introduced. They learned the spatio-temporal patterns of human activities, estimating the waiting time and location for activities. Using Hidden Markov Models they predicted motion of people to track them and adapt the robot's navigation behavior in a simulated office-like environment.

When navigating in highly crowded areas, the path planning should take the people's movement into account. In [16], the robot was able to navigate in an unknown urban environment that furthermore was highly populated. In the experiments, they led their robot navigate alone without knowing the map by asking people for directions. The robot used Simultaneous Localization and Mapping (SLAM) and was able to follow and interact with people. Trautman et al. [17] proposed a method for a mobile robot to navigate in dense crowds, in which they intended people to cooperate with the robot to avoid collisions and reach its goal. They used Gaussian processes to model the motions of other agents, and they planned a path to go from one place to another taking into account the probable paths of the other agents. They found the method worked better than a standard planner that does not take into account other people's paths, and the performance was comparable to a human teleoperated robot when the density is 1 person/m^2. A blind guiding robot, created by [11], used a Kinect camera to detect if the person was following. It used a

Gaussian Mixture Model (GMM) for the person, and another for the obstacles, which were trained by hand-classified objects. Their system was able to detect the person even when some occlusions occurred.

Tracking can also be done with multiple fixed sensors placed in the zone where people should be tracked. In [18], a particle filter was used to track the people; they used multiple fixed 3D sensors, which where mounted on the top, to reduce occlusions. Experiments done in a shopping center showed good results, but they had some problems with unexpected obstacles and lightening conditions. Other work has been done on Wireless Sensor Networks (WSNs), in which a large amount of connected sensors is used for tracking. In [19], they used Distributed Particle Filters (DPFs) to track objects or persons, and they presented a method to have higher computation and communication efficiency.

In [20], we focused on the hide-and-seek game, because this gives a limited and easier working environment to study the problem of finding people. First, we started to work in discrete time and space [21], and we used the Reinforcement Learning model MOMDP (Mixed Observable Markovian Decision Process) [22] to search for a person. In [4], we proposed a method for large continuous state space, the *Adaptive Highest Belief CR-POMCP Follower (Adaptive HB-CR-POMCP Follower)*, which uses the probability map (belief) that is updated by the CR-POMCP (Continuous Real-time Partially Observable Monte-Carlo Planning) method. CR-POMCP requires a POMDP (Partially Observable Markovian Decision Process) simulator $(s', o, r = \mathcal{G}(s, a))$ to be defined, which returns the next state, observation, and reward based on a current state and action. Observations are used since the state is not fully observable, because the sensors are noisy and the person might not always be visible. Therefore, to keep track of the state, a probability map (the belief) is maintained. The belief is used to search for the person, since the use of the actions of the policy caused several practical issues, as discussed in [4]. In order to decide where the robot should go to, the location of the highest belief (HB) is calculated, and the robot is navigated there using, for example, a Dijkstra planner. Given that the CR-POMCP uses continuous states, we needed to group belief points in order to find the location with the highest belief. This is done by using a grid of equally sized squares, because it is a quick method, and most clustering methods are too complex. Finally, when the person is visible, the robot tracks him/her using the person's location to avoid imprecision due to the discrete belief grid.

In this work we present the method *Highest Belief Particle Filter Searcher and Tracker*, which is based on the particle filter, but shares the properties of updating the belief with the previously presented *Adaptive HB-CR-POMCP Follower* [4]. The CR-POMCP uses belief points which are similar to the particles of the particle filter. The main difference is the way the belief is updated, in the CR-POMCP a POMDP simulator and the policy update method are used; whereas the presented method only requires to redefine the weight function of the particle filter. Furthermore, here we present a method that takes the dynamic obstacle into account while searching and tracking. Our method works on-line and is able to both search and track a person, in contrast to trackers as [12–14]. Moreover, we use a mobile robot instead of multiple fixed sensors like [18,19]. Whereas [6,7] focus on multi-agent coordination and assume full visibility in the map's vertices, we present experiments done in real-world urban areas with a continuous state space. Finally, the work of [5] assumes the person not to move during the search whereas in our method the person can move.

3. System architecture

The complete architecture of the *search-and-track* system, which we present in this paper, is shown in Fig. 2. The architecture shows the components that are used to search and track, where our contributed components are shown in bold.

In order to search and track, we first need to know the positions of the robot and the visible people, which is done in the *Robot Perception* modules. The *Robot Localization* (Section 5.3) uses the *Odometry* and *Laser* sensors to localize itself and returns the agent's position $O_{\text{L,agent}}$. To detect all the people in the visible field, the *People Detection* algorithm uses the horizontal range *lasers*, which detects leg shaped objects. Next, a tracking algorithm is used to keep track of the detections while they are visible. The list of detected people is then used as *dynamic obstacles* in our search-and-track algorithm, and in the *Person Localization* block. To recognize the person we are looking for, we put an AR Marker [23] on the person, see Fig. 1. The *Person Recognition* module detects the AR marker and outputs its positions, which is then combined with the location of all the detected people in the *Person Localization* to generate the observed location of the person $O_{\text{L,person}}$. Note that this observation can be empty if the person is not detected. The *Observation Filter* makes sure that the observed robot location and person locations are inside the map, and that they are not inside an obstacle.

Next, the *search-and-track* method is executed using the observations. First it starts by updating the belief (a probability map) of the person's location using a particle filter with a modified weight update function, as explained in Section 4.2. Then, the robot goal is selected (see Section 4.3), which depends on the observation of the person; if he/she is visible, the robot does tracking using the observed position, otherwise *search* is done. The *search* uses the belief to calculated the location with the highest belief, which is then selected as goal.

Finally, the *Robot Navigation* module calculates the path to the selected goal, which is sent as motion commands to the robot.

4. Highest belief particle filter searcher and tracker

Our goal is to search and track a person in an urban environment that has static and dynamic obstacles. We have already seen in the related work that there exist different good techniques for searching or tracking only. In our work we present a method that is based on the particle filter, but it can do searching and tracking using the same algorithm.

Searching is done when the person is not visible, and for this we use a probability map, the belief, to estimate the person's position. In [4], we used a CR-POMCP to update the belief, and in this paper we present an adapted particle filter to maintain the belief. By creating a 2D histogram of the belief, we are able to find the area with the Highest Belief, which is the most probable location of the person. Next, the robot is sent to this location, and this process is repeated – updating the belief and robot goal – until the person is found.

Tracking is following the person when he/she has been visible to the robot. This is done using the observed location if available, but if the robot loses person due to occlusion, distance, or sensor noise, the Highest Belief is used, like in the *search* phase.

With this, we have created a uniform method which can both *search* and *track*. Furthermore, the method is improved by also taking into account the dynamic obstacles when estimating the location of the person. This results in a method which not only looks behind static obstacles, but also behind other people walking around for example.

In this section, we will first explain the basic particle filter, then the extensions made to maintain a belief of the person's location in Section 4.2. Then the next section explains how the Highest Belief

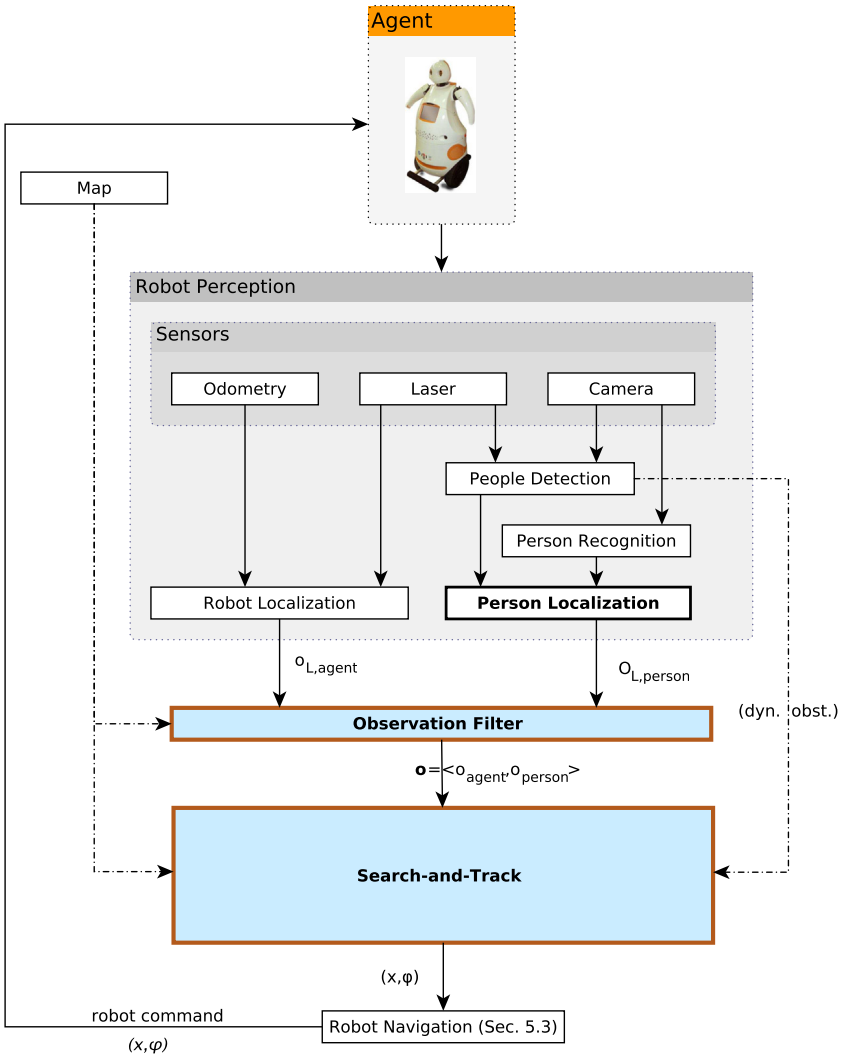

Fig. 2. The architecture of the *search-and-track* method presented in this paper with our contributions shown in bold, and the sections in which the items are discussed are shown between parenthesis.

is calculated, and as last is explained how the dynamic obstacles are used in the prediction.

4.1. Basic particle filter

A particle filter can be used to track the state of a robot [24], or a robot and a person [25]. In our case we track and search for the person, and therefore the state is defined as the position of the person: $s = (x, y)$. Since the exact state (i.e. position of the person) is not known, it is estimated by using a large number of particles. A basic particle filter is shown in Algorithm 1. The particles are first propagated in line 3 according to $p(s_t|s^i_{t-1})$, then the weight of particle i is calculated in line 4, and after this, resampling is done in lines 7–10.

Algorithm 1 A basic particle filter.

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

4.2. Modifications of the basic particle filter for search-and-track

To make the particle filter suitable for searching, we have made two important changes. First, the initial distribution is based on the initial observation, taking into account the locations where the person could be hidden. Second, the lack of observation to update the particles is also used.

The initialization of the N particles is done randomly based on the observation $o = (o_{\text{agent}}, o_{\text{person}})$, as shown in Algorithm 2. If the person is visible then its position is used, otherwise a random position in the map that is not visible to the seeker is chosen. After that, Gaussian noise is added to the position of the person in line 9.

In the prediction step we assume the person to have moved according to a Gaussian (with standard deviation σ_{person}) in a random direction (θ):

$$s_t = s_{t-1} + \mathcal{N}(1, \sigma_{\text{person}})[\cos\theta, \sin\theta]^T. \quad (1)$$

Algorithm 2 Initialization of the modified particle filter.

```
1: function INIT(o,N)
2:     S = ∅
3:     for i = 1 to N do
4:         if o_person = ∅ then                ▷ i.e. person not visible
5:             s = RANDOMNOTVISIBLEPOS(o)
6:         else
7:             s = o
8:         end if
9:         s = N(s, σ_person I)
10:        S = S ∪ {s}
11:    end for
12:    return S
13: end function
```

Algorithm 3 Changed update step that also takes into account cases where there is no observation, using $w(s, o)$, see Eq. (2).

```
1: function UPDATE(o,S,N)
2:     ∀_{s∈S} : s_w = w(s, o)
3:     ∀_{s∈S} : s_w = s_w / Σ_{k∈S} k_w          ▷ normalize
4:     S̄ = ∅
5:     for i = 1 to N do
6:         s̄ sample from S                       ▷ with probability s̄_w
7:         S̄ = S̄ ∪ {s̄}
8:     end for
9:     S = S̄
10: end function
```

The update step was modified, see Algorithm 3, such that the particle weight is changed, even if no observation is available of the person:

$$w(s,o) = \begin{cases} 0, & \text{if } \neg \text{ISVALID}(s) \\ e^{-|o_{\text{person}} - s|^2/\sigma_w^2}, & \text{if ISVALID}(s) \wedge o_{\text{person}} \neq \emptyset \\ w_{\text{cons}}, & \text{if ISVALID}(s) \wedge o_{\text{person}} = \emptyset \\ & \quad \wedge P_{\text{vis}}(o,s) = 0 \\ w_{\text{inc}}(1 - P_{\text{vis}}(o,s)), & \text{otherwise} \end{cases} \quad (2)$$

where ISVALID checks if the state is a valid free position in the map. If the person is visible, a probability is calculated based on the distance between the observed location and the particle location, where for a higher σ_w higher distances are accepted (we set $\sigma_w = 1.0$). If the person is not visible, then we cannot measure the error of the particles (distance), but we can only check if the particle should be visible or not to the agent, and therefore is at least consistent or not. If the particle is not visible, it is consistent with the observation, and we give it a constant weight w_{cons} (0.01). However, if there is a probability of seeing the particle $P_{\text{vis}}(o, s)$ (Eq. (3)), then a lower weight $w_{\text{inc}} \ll w_{\text{cons}}$ is used (we set it to 0.001). The visibility probability is defined as the following piecewise linear formula:

$$P_{\text{vis}}(o,s) = \begin{cases} 0, & \text{if RAYTRACE} \\ \quad (o_{\text{agent}}, s) \text{ not free} & \\ p_{\text{v,max}}, & \text{if } d < d_{\text{v,max}} \\ \max(0, p_{\text{v,max}} - \alpha_{\text{vis}}(d - d_{\text{v,max}})), & \text{otherwise} \end{cases}$$

where RAYTRACE is used to check the visibility, and $d = \|s - o_{\text{agent}}\|$. The constants for our robot setup were estimated based on the real experiments, and resulted in: $d_{\text{v,max}} = 3.0$, $p_{\text{v,max}} = 0.85$ and $\alpha_{\text{vis}} = 0.17$.

4.3. Highest belief calculation to estimate the person location

The particle filter method, until now, gives a probability map of the person's location. Like discussed previously and in [4], we decided to use a grid to find the highest belief. To prevent the agent from changing too quickly from one point to another, the goal is maintained during several time steps (3 steps in simulation and 3 s during the experiments). In larger maps the belief grid cells should not be too small in order to accumulate enough particles. Also a maximum search distance of 25 m ($d_{\text{max\_search}}$) was set in order to have the robot not go too far; only in the case of not finding any highest belief in this area, the search space was increased to the whole map.

Like in [21], the observed location by the robot is used when the person is visible such that the precision is higher and we do not depend on the grid.

4.4. Extension of the method to handle dynamic obstacles

In our previously presented method [21], we predicted the possible locations of the person based on the known map of the environment. For the method to work with dynamic obstacles, we added false negatives to the prediction phase. In this work, however, we also take into account dynamic obstacles that can occlude the person. For this, we have also taken into account the detected dynamic obstacles when executing the RAYTRACE function.

The use of dynamic obstacles in the prediction of the particle locations helps to prevent falsely not detecting the user. In the case of not taking into account dynamic obstacles, the system assumes – after not detecting anything – that the area in its surrounding is free. When dynamic obstacles are taken into account, particles behind them are given a higher weight w_{cons}, see Eq. (2). Handling false positive detections is more difficult, but they can be partly treated by the particle filter method. Several iterations are needed in which the incorrect observation is detected in order to concentrate all the particles to that location. In a worst-case scenario where most particles are concentrated at the incorrect location they are propagated away from it as soon as the incorrect detection is lost.

5. Experimental setup

In this section the experimental setup is explained: first we give some details about the used mobile robot and the used algorithms to localize the robot and person; next, the environments in which the experiments were done are commented.

5.1. The robot

For the experiments we have used our mobile service robot Dabo, which has been created during the URUS project [26], together with its twin Tibi. They were designed to work in urban pedestrian areas and to interact with people. Tibi and Dabo 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, which leaves a blind zone of about 47 cm on each side. A PointGrey Ladybug 360° camera located on the top of the head is used for computer vision purposes.

As social robots, Tibi and Dabo (see Fig. 1) are meant to interact with people, and to perform this they have: a touchscreen, a 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 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. An external laptop (Intel Core i5-2430M @ 2.40 and 3.00 GHz with 4 GB RAM) is used to run the search-and-track algorithm, and for external monitoring. As operating system the systems run Ubuntu 14.04 with ROS (Robot Operating System), a middleware.

5.2. People recognition and dynamic obstacles

Since our algorithm uses as input the location of the person, if visible, we make use of several already existing algorithms that give us this information using several sensors, as shown in Fig. 2.

A boosting leg detector [27] provides the position of potential people in the scene using the horizontal front and rear range laser sensors. A Multiple Hypothesis Tracking for Multiple Targets [28] (MHT) keeps the trail of the detected people.

A people detection algorithm alone is not enough, because we also have to recognize the person we are looking for. Although some works have used people detection, they require either a large amount of computing time [5] or a 3D camera such as the Kinect [11], which does not work well outdoors. Therefore, we decided to use a robust method, AR Markers (Augmented Reality Markers) [23], which were worn by the person, see for example Fig. 1. The AR algorithm gives an estimation of the pose of the AR marker with respect to the camera. We use an improved version of this Pose Estimation algorithm [23] that, in combination with previous local window binarization, makes the method more robust to outdoors lighting issues. We use the Ladybug 360° camera (which internally has five cameras) to detect a tag from any direction. The AR detection algorithm is run on one computer for all five cameras of the Ladybug and ran on average at 4 Hz.

Since the AR marker detector sometimes gives false positive detections, we decided to also use the list of detected people returned by the MHT algorithm, and only if the AR Marker detection is close to the location of a detected person, it is accepted. As side-effect, some false negatives occur, but these had a lower effect on the search-and-track method because in most cases they just delay the detection of the person whereas a false positive detection misguides the robot to go to an incorrect place.

The locations of the detected people are also used in our presented algorithm as dynamic obstacles, which are then used to model the possibility of the person being behind the dynamic obstacle, as has been explained in Section 4.4. Finally, the *Observation Filter* tries to correct the locations of observations in an obstacle or outside the map [4].

5.3. Robot mapping and navigation

As can be seen in Fig. 2, our search-and-track algorithm requires the position of the robot on the map, and requires an algorithm that navigates the robot to a *goal* position. Although the robot localization and tracking of people can be done simultaneously [25], we focus on the search-and-track of a person, and therefore we make use of existing algorithms that do localization and navigation of the robot.

Prior to the experiments a map has been 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 filter that learns grid maps from laser range data [29]. 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 too long. Instead, we use 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 [30].

The robot moved through the environment using a set of navigation algorithms provided by ROS. It consists of a Dijkstra global planner which uses the previously generated map to calculate the shortest path. To avoid dynamic obstacles, a local Trajectory Roll Out [31] planner is used, which generates and scores trajectories over a costmap that is updated with range laser data. The inputs of the navigation algorithm are the desired goal coordinates and orientation, see Fig. 2, which are given by our search-and-track algorithm.

5.4. Environments and maps

Since there are no standard search-and-track data sets, we have used a large environment (*Telecos Square* Fig. 3(b) and (c), 60 m × 55 m of which 1400 m^2 is accessible), which represent diverse pedestrian urban environmental types, and a smaller environment, the FME (Facultat de Matemàtiques i Estadística lab; 17 m × 12 m, 170 m^2 accessible). Both outdoor urban environments are located respectively at the North and South Campus of the Universitat Politècnica de Catalunya (UPC), Barcelona, Spain.

The *FME* environment is outdoor, but partly covered by a roof, and since no obstacles were in the field we placed several artificial ones, as can be seen in Fig. 3(a). The *Telecos Square* is the largest environment which contains a square, and two covered areas with several columns. Through both areas people pass by frequently, especially the last since it is the center of the campus.

6. Simulations and real-life experiments

To verify the methods, first we did simulations, and thereafter real-world experiments were done in urban environments.

6.1. Simulation

The maps contain discrete cells which either are free or obstacles, where obstacles indicate that neither robot nor person can pass through it, nor can see through it. A ray tracing algorithm is used in simulation to detect visibility. 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 $\sqrt{2}$). The vision of the robot is limited due to occlusions by obstacles and the maximum visibility distance. The latter is estimated using real experimental data, like explained in Section 4. The visibility probability Eq. (3) is also used to simulate observations, i.e. an observation with the real person's location is returned with a probability of $P_{\text{vis}}(o, s)$, otherwise an empty observation is returned. The simulations do not include neither acceleration, nor friction, nor collision, for simplicity. Furthermore, the agents are not allowed to be neither outside the map nor on top of an obstacle.

The algorithms calculate a goal, and in the simulator the agent is moved one step at a time in the goal's direction using the shortest path. The persons are simulated by giving them goals to go to. They start at a random position with a random goal. In each iteration, the goal is approached one step, and when the goal is reached, a new random goal is chosen.

A noisy crowded environment has been simulated by adding groups from 10 to 100 people to the scene. These people moved, as the person to be found, to randomly selected goals. By doing this, they reduce the robot's visibility.

The tested algorithms are the *HB-PF Searcher and Tracker*, the *Adaptive HB-CR-POMCP Follower*, both with and without the use of dynamic obstacles. As a reference method, we added a following seeker which always sees the person, the *See All Follower*, independent of its distance and obstacles.

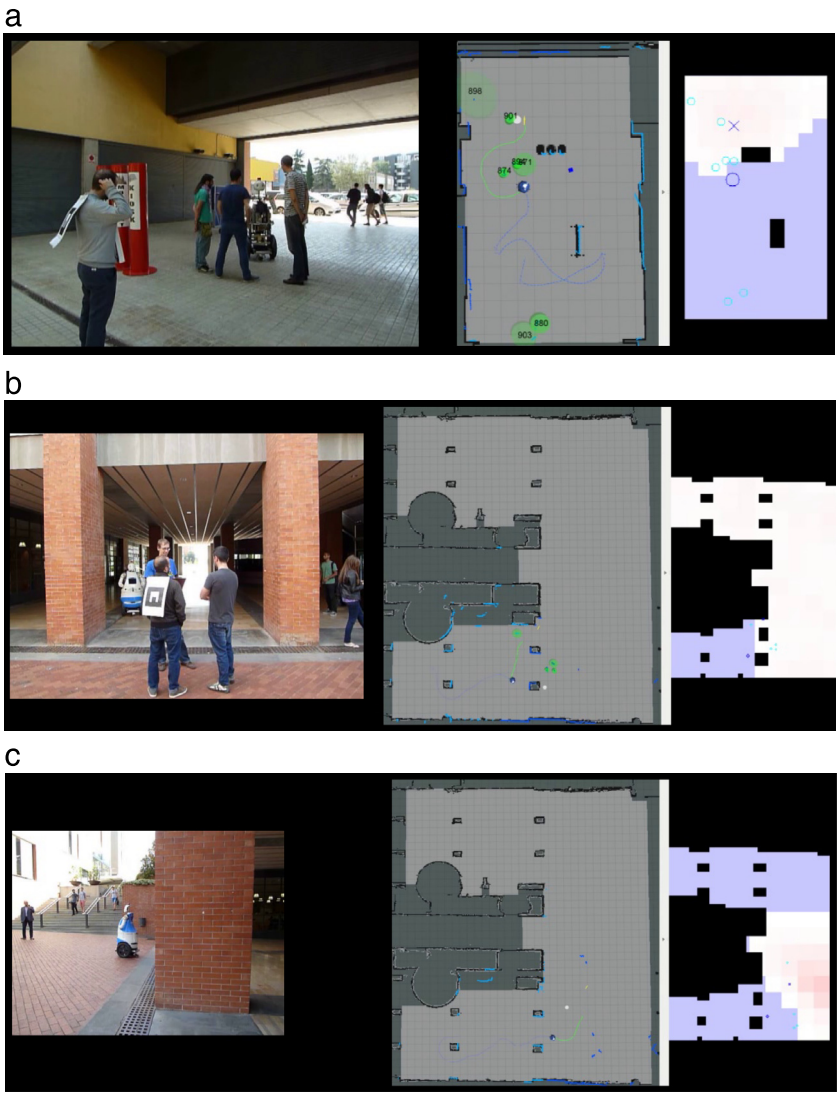

Fig. 3. The FME map (a) with a size of 17 m × 12 m, and an accessible surface to the robot of about 170 m^2. The Telecos Square map (b, c) with a size of 60 m × 55 m of which about 1400 m^2 is area accessible to the robots. The center image shows the map used for localization with the robot (blue), detected people (green), and the laser range detections. The right image shows the probability map, i.e. belief, of where the person could be; here red indicates a high, white a low, and light blue zero probability. The blue circle indicates the location of the robot, the light blue circles are the locations of other nearby people, and the cross is the robot's goal. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

More than 8000 experiments have been done, repeating each of the conditions at least 140 times. For each run of simulations, the robot's start position, and the person's start and end position were generated randomly. To make the comparison as fair as possible, the same positions were used for the five tested algorithms, such that the initial state and the person's movement were the same, which was repeated several times for the same conditions.

The experiments are separated in *search*, and *track*. In the first, the person starts hidden from the seeker and stays there, and the simulation stops when the seeker finds the person and is close to it. Simulations are stopped if the maximum time passed: 500 steps for the FME map, and 5000 for the Tel. Square map. Here we measure the time (steps) the seeker needs to see the person. In the track experiments, the seeker starts close to the person and 500 steps are done for the FME map, and 1000 for the Tel. Square map. Here we measure the distance to the person, the time of visibility, and the recovery time, which is the time the agent needs on average to see the person again.

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

$$\varepsilon_b = \sum_{x \in A} b_x \|x - p\| \tag{4}$$

where A is the discrete map, x represents a grid cell, and b_x is the belief of cell x. The average of this value is used to compare the beliefs.

6.2. Algorithm parameter values

The values of the parameters used in the simulations and real experiments are shown in Table 1, and were obtained experimentally. The algorithms update their particles or belief every iteration, and the highest belief points are calculated every 3 s in the real experiments and every 3 iterations in the simulations if the person is not visible.

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

Parameter	FME	Tel.Sq.
N	1000	5000
σ_{person} (m)	0.3	
w_{cons}	0.01	
w_{inc}	0.001	
σ_w (m)	1.0	
HB cell size (m × m)	1 × 1	3.8 × 3.8
Belief update time[a]	3 s; 3 steps	
$d_{\text{max\_search}}$ (m)	10	25

[a] For simulation we use discrete time steps, for the real experiments seconds are used.

6.3. Results

The results of the simulations, the 140 repetitions, are shown in Table 2 and 3 for the FME and Tel. Square map respectively. We did *search* and *track* simulations separately, the measurements in time are discrete time steps, and for the distance, the cells are converted to meters. The cell size for the FME was 1.0 m and for the Tel. Square 0.8 m, and the average speed was 1 cell per time step. Although the *See All Follower* was always given the location of the person, we do use the real visibility (i.e. taking into account distance and obstacles) when we calculate the measurements (*First visible step*, *Visibility*, and *Recovery time*), which gives us the best possible values for the measurements.

In the *search* phase the *first visible step* (the discrete time until the person was found) was measured, for which only a significant difference was found with the *See All Follower*. The high standard deviation is due to the large difference in the starting position of the robot and person for the simulations. The distance to the person during the *tracking* phase was found to be significantly less for the HB-PF Searcher & Tracker ($p < 0.001$; Wilcoxon Ranksum test) in comparison with the Ad. HB-CR-POMCP Follower, except for in the FME map without people walking around. When looking at the belief error Eq. (4), there is no clear difference between the methods.

The visibility was found to be significantly higher ($p < 0.001$; Fisher's exact test) for the HB-PF Searcher & Tracker, except for the case of 100 random people walking around. Finally, for the *tracking* phase, the *recovery time* shows the average time the agent needs to recover the visibility of the person after having lost him/her. It shows that after the *See All Follower*, the HB-PF S.&T. works best for both maps ($p < 0.01$; Wilcoxon Ranksum test) without dynamic obstacles. With dynamic obstacles the *See All Follower* had better results, but there is no significant difference between the other methods. For the HB-PF Searcher & Tracker algorithm, the use of the detected dynamic obstacles in the algorithm had a positive effect on the *distance to the person* when tracking, on the bigger map ($p < 0.01$; Wilcoxon Ranksum test). Also for the *visibility* this had a positive influence.

The runtime of the algorithms is not significantly different, for search the average was about 250 ms/iteration, and for track about 216 ms/step. Note that the run time mainly depends on the number of particles for the HB-PF Searcher & Tracker or on the number of belief points for the Ad. HB-CR-POMCP Follower.

Using the dynamic obstacles, which have been detected to update the probability map, has a great advantage in certain situations, as shown in Fig. 4. Left can be seen the situation where there are particles maintained behind the detected dynamic obstacles, whereas in Fig. 4(b) that area is already cleaned. In this simulation the first method needed 21 steps to find the person, whereas the latter needed 366 steps, because it went around the obstacle, thinking that the person could only be hidden behind obstacles. See a demonstration video on http://www.iri.upc.edu/groups/lrobots/search-and-track/ras2016/.

6.4. Real-life experiments

The real-life search and track experiments have been done with our mobile robot Dabo, using the algorithm explained in this work, *HB-PF Searcher and Tracker*. In the FME environment three experiments were done and all of them were successful. The total robot movement was 160 m, the total person movement was 35 m (when visible) and the total experiment duration was 702 s. In the Tel. Square environment 18 experiments were done, of which 11 were successful (the robot was able to search and track the person), and in 7 experiments several problems occurred. The issues were not related to the search-and-track method, but to software failure or hardware problems (e.g. temperature or low battery) which stopped the robot. In the 11 experiments the robot moved 1173 m, the person moved 148 m (when visible) and the total duration of the experiments was 5925 s (more than 1.5 h). Table 4 shows an overview of all the experiments' statistics, which are based on the data measured by the robot's sensors, so the distance traveled by the person was more than the values reflected in the table. The videos of the experiments and more information can be found on http://www.iri.upc.edu/groups/lrobots/search-and-track/ras2016/.

We were not able to get the same statistics as the simulations (Table 2–3) for the real-life experiments, due to two main reasons. First, the ground truth should be known to have correct evaluation measurements like in the simulations, and since most global person localization methods have a low precision, we decided not to use any. Instead we have used the person location obtained with the leg and AR marker detectors (see Section 5.2), which allow us to calculate the *distance to the person* when the person is visible. Second, several problems during the experiments caused the robot to stop a few seconds, extending therefore the time measurements. Nevertheless, we were able to show some descriptive statistics, as shown in Table 4, and we were able to obtain more specific statistics for the *search* and *search-and-track* experiments shown in Table 5.

For the *search* experiments we calculated the time to first seeing the person; the tracking is counted from when the person is found, and in this case, we measure the average distance to the person, visibility, and the recovery time. Note that the *Belief Error* cannot be calculated since it requires us to know the ground truth.

6.4.1. Smaller map—FME

The FME map Fig. 3(a) was slightly easy for the robot since the area is relatively small, however, there were always two fixed obstacles where the person could hide. Furthermore, it can be seen in Fig. 3(a) that the people standing in front of the robot also were used as obstacles, because there was a belief (i.e. particles) behind them.

In Table 5, for the FME, we only show one experiment because this was a complete search-and-track experiment, and in this experiment there were four people walking around. We can see that the distance to the person and also the recovery distance (comparing it to the recovery time) are comparable to the results of the simulation (Table 2).

6.4.2. Bigger map—Telecos Square

In the bigger map, the robot needed more time to explore the environment, because of its size and the relatively low speed of the robot. Compared to the previous method [4], our search-and-track method also includes searching in presence of dynamic obstacles, which has been shown to be important in several experiments. In various experiments, the robot looked for the person behind one or more dynamic obstacles, such as for example in Fig. 3(b), where three people in front occlude part of the map which otherwise would be visible to the robot.

Table 2
This table shows the average and standard deviations of the simulations in the **FME**. The different algorithms are shown in columns, where *(d)* indicates the use of dynamic obstacles. The type of measurement is indicated in the first column, including the used unit and the type of task (search or track). The *First visible step* is the first time (discrete time) when the person was visible to the seeker. The *visibility* indicates the amount of time (% of the whole simulation) in which the person was visible to the seeker. The *distance to the person* is the average distance to the person, and the *belief error*, Eq. (4), indicates how well the belief represents the real location of the person. The *recovery time* shows the average time the agent needed to return to see the person after having lost him/her.

Type of measurement	Num. Pers.	*See All F.*	*HB-PF S&T* (d)	*HB-PF S&T*	*Ad.HB-CR-POMCP F.* (d)	*Ad.HB-CR-POMCP F.*
First visible step	0	2.4 ± 3.2	4.5 ± 5.8	4.7 ± 8	5.3 ± 6.9	5 ± 6.5
(time steps)	10	3.1 ± 3.7	7.2 ± 9.1	6.9 ± 10.7	7.7 ± 11.8	7.4 ± 10.1
(search)	100	8.4 ± 6.4	67.4 ± 71.1	90 ± 111.1	65.2 ± 78.8	59.9 ± 67.5
Visibility	0	100%	93.7%	93.4%	94.5%	93.9%
(%)	10	100%	59.3%	56.5%	59.5%	58.4%
(track)	100	100%	2.4%	1.8%	2.8%	2.8%
Distance to pers.	0	1.0 ± 0.4	2 ± 1.4	2 ± 1.4	1.9 ± 1.5	1.9 ± 1.5
(m)	10	1.0 ± 0.4	3.2 ± 2.5	3.4 ± 2.6	3.3 ± 2.8	3.5 ± 2.9
(track)	100	1.0 ± 0.4	5.6 ± 2.9	5.4 ± 2.8	6 ± 3.2	6 ± 3.2
Belief error	0		4.2 ± 3	4.4 ± 3.3	5.1 ± 3.1	5 ± 3.2
(ϵ_b) (m)	10		5.1 ± 3.2	5.4 ± 3.4	6 ± 3.1	5.7 ± 2.9
(search)	100		7.4 ± 1.9	8 ± 3.4	7.7 ± 1.8	7.5 ± 1.8
Belief error	0		1.1 ± 1.1	1.1 ± 1.1	0.8 ± 1.3	0.9 ± 1.4
(ϵ_b) (m)	10		2.5 ± 2.5	2.8 ± 2.7	2.5 ± 2.9	2.6 ± 3
(track)	100		6.4 ± 1.9	6.3 ± 2.9	6.6 ± 2	6.7 ± 2
Recovery time	0	1.2 ± 0.5	2.4 ± 1.9	2.5 ± 2.1	2.4 ± 3.2	2.7 ± 3.8
(time steps)	10	2.2 ± 3.8	3.5 ± 4.4	3.9 ± 4.6	3.6 ± 5.2	3.8 ± 5.5
(track)	100	3.2 ± 5.7	48.8 ± 54.5	63 ± 67.2	46.5 ± 55.8	46.7 ± 53.9

The *recovery time* is identical to the *recovery distance* since the agent moves 1 m per time step on average.

Table 3
This table shows the same measurements as Table 2, but for the **Tel. Square** map.

Measurement	Num. Pers.	*See All F.*	*HB-PF S&T* (d)	*HB-PF S&T*	*Ad.HB-CR-POMCP F.* (d)	*Ad.HB-CR-POMCP F.*
First visible step	0	13.0 ± 16.2	110 ± 336.2	106.2 ± 369.6	114.6 ± 264.3	110.4 ± 233.2
(time steps)	10	13.8 ± 21.7	105.6 ± 285.5	96.5 ± 273.8	93.7 ± 189.3	107.4 ± 237.1
(search)	100	14.4 ± 17.2	115.6 ± 264.9	127.7 ± 265.9	121.2 ± 300.5	117.5 ± 246.9
Visibility	0	100%	62.7%	61.6%	58.5%	60.4%
(%)	10	95.6%	51.2%	45.6%	48.8%	49.0%
(track)	100	70.5%	13.8%	12.8%	15.9%	15.7%
Distance to pers.	0.0	0.8 ± 0.4	8.2 ± 9.1	8.6 ± 9.5	8.5 ± 9.0	8.3 ± 9.1
(m)	10.0	0.8 ± 0.4	9.4 ± 9.4	11.0 ± 10.4	9.8 ± 9.6	9.6 ± 9.4
(track)	100.0	0.8 ± 0.4	13.6 ± 9.4	15.4 ± 10.4	13.8 ± 9.7	13.8 ± 9.6
Belief error	0.0		25.4 ± 11.8	26.1 ± 9.4	23.8 ± 6.9	23.4 ± 6.5
(ϵ_b) (m)	10.0		25.9 ± 10.0	26.5 ± 9.8	23.0 ± 6.9	23.4 ± 7.1
(search)	100.0		25.4 ± 9.3	25.2 ± 9.2	23.2 ± 6.2	23.3 ± 6.9
Belief error	0.0		7.5 ± 10.1	7.8 ± 10.2	7.7 ± 9.4	7.4 ± 9.6
(ϵ_b) (m)	10.0		9.0 ± 10.2	10.8 ± 11.3	9.2 ± 10.0	9.0 ± 9.9
(track)	100.0		14.9 ± 9.2	16.7 ± 10.4	14.1 ± 9.3	14.2 ± 9.2
Recovery time	0	1.2 ± 0.5	14.9 ± 31.6	15.3 ± 32.5	19.5 ± 34.2	19.1 ± 34.8
(time steps)	10	2.2 ± 3.8	13 ± 27.9	15.6 ± 35.5	15.4 ± 31.6	15.5 ± 32.2
(track)	100	3.2 ± 5.7	22.2 ± 42.7	24.8 ± 48.6	19.7 ± 41.2	20.1 ± 41.9
Recovery dist.	0.0	1.0 ± 0.4	11.9 ± 25.3	12.2 ± 26.0	15.6 ± 27.4	15.3 ± 27.8
(m)	10.0	1.8 ± 3.0	10.4 ± 22.3	12.5 ± 28.4	12.3 ± 25.3	12.4 ± 25.8
(track)	100.0	2.6 ± 4.6	17.8 ± 34.2	19.8 ± 38.9	15.8 ± 33.0	16.1 ± 33.5

In some occurrences of false positive detections, the robot stayed longer than required, because the robot assumed that the person was near to it. Then, the robot recovered either because it detected the real person or the false positive detection disappeared. The false detections were because other person's legs were detected in combination with a falsely detected AR marker.

In Table 5, we include for the real-life experiments the same measurements that we did in simulation. The *first visible step* should be compared (between simulation and real-life experiments) using the distance and not the time, since the robot stopped several times without moving due to external problems. Since in the real experiments there were around 5–10 other people walking around in the scene, we should compare it to the 105.6 ± 285.5 time steps in simulation, which is about 84.5 ± 228.4 m (0.8 m/cell). In the real experiments these values were lower which might be because only a few search start positions were tried, whereas in simulations many completely random positions were tried. The visibility (in the tracking phase) was lower in the real experiments, mainly because the robot's vision system is worse than in the simulated version, which for example assumes a full panoramic view; note that in Table 4 the visibility was very low, because it includes exploration experiments in which the person was not present. The average distance to the person when tracking for the simulations was 7.5 ± 7.5 m, which is more than in the real world, because there it could only be calculated when the person was visible. The lower real world recovery distance could be explained by the low speed of the person.

The real experiments showed us that the HB-PF Searcher & Tracker is able to search and track the person, also when other people are partly blocking the field of view.

7. Conclusion

In this work we have presented a unified method that uses a modified particle filter: *Highest Belief Particle Filter Searcher and*

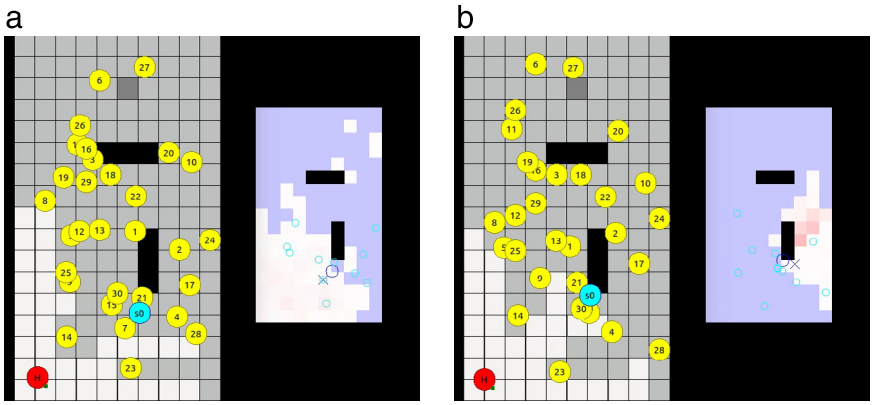

Fig. 4. (a) The simulated seeker using dynamic obstacles, and (b) not using dynamic obstacles. The left image of the image pair shows the person as red circle, the blue circle as the robot, and yellow circles are other people walking around. The black cells are obstacles, light gray are cells visible to the person, and dark gray are not visible cells. The right part shows the probability map, i.e. belief, of where the person could be where red is a high probability, white low, and light blue zero. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 4
Summary of the real world experiments.

Measurement	FME	Tel.Sq.
Average distance to person[a] (m)	2.8 ± 1.4	4.2 ± 2.4
Person visible time (%)	19.6%	3.2%
Average number of dyn. obstacles	4.0 ± 1.7	3.2 ± 2.4
Max. number of dyn. obstacles	8	16

[a] When the person is visible to the robot, because we used the robot's sensors.

Table 5
Here the average values are shown for the real world experiments.

Measurement	FME[b]	Tel.Sq.
First visible time (s) (search)	4	132.5 ± 84.8
First visible distance (m) (search)	0.35	37.6 ± 23.9
Visibility (%) (track)	25.2%	16.0%
Distance to pers.[a] (m) (track)	2.9 ± 1.4	4.2 ± 2.5
Recovery time (s) (track)	17.1 ± 18.6	22.9 ± 40.0
Recovery dist. (m) (track)	4.0 ± 5.5	4.3 ± 6.6
Robot movement/experiment (m)	105.5	62.5
Person movement[a]/experiment (m)	32.0	13.2

[a] Only person locations are used of measured points, i.e. when visible to the robot.

[b] The data for the FME has only one experiment.

Tracker that is able to autonomously search and track a person in an urban environment. The method is able to work without having an observation, with false detections of the person, under uncertainty, in continuous space, and in real-time. Furthermore, we have presented a method to take into account dynamic obstacles while searching and tracking a person. This, as expected, improves the search in crowded areas, because now it maintains a probability of the person being behind any of the dynamic obstacles. To choose the goal, the particles are accumulated in square areas, and the area with the highest concentration (*Highest Belief*), within a certain maximum range, is chosen.

Comparing the *Highest Belief Particle Filter Searcher and Tracker* method with the *Adaptive HB-CR-POMCP Follower* [4] method (that we previously developed), the current method is much simpler, because it is much easier to modify the algorithm strategy in the particle filter than in the POMCP, even though the simulation results are similar.

As future work the robot's path should be taken into account while doing searching, furthermore, a path prediction method of the person could be used when tracking, such as [32]. Finally, for a more pragmatic system, a person recognition system should be used that does not require artificial markers.

Acknowledgments

This work has been partially funded by the EU project AEROARMS European project H2020-ICT-2014-1-644271 and the CICYT project DPI2013-42458-P.

References

[1] K. Dautenhahn, S. Woods, C. Kaouri, M.L. Walters, K.L. Koay, I. Werry, What is a robot companion-friend, assistant or butler? in: Proceedings of the IEEE International Conference on Intelligent Robots and Systems, IROS, 2005, pp. 1192–1197.
[2] B. Doroodgar, M. Ficocelli, B. Mobedi, G. Nejat, The search for survivors: Cooperative human-robot interaction in search and rescue environments using semi-autonomous robots, in: Proceedings of the IEEE International Conference on Robotics and Automation, ICRA, 2010, pp. 2858–2863.
[3] A. Garrell, M. Villamizar, F. Moreno-Noguer, A. Sanfeliu, Proactive behavior of an autonomous mobile robot for human-assisted learning, in: Proceedings of IEEE RO-MAN, 2013, pp. 107–113.
[4] A. Goldhoorn, A. Garrell, R. Alquézar, A. Sanfeliu, 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, 2014, pp. 741–747.
[5] M. Volkhardt, H.M. Gross, Finding people in apartments with a mobile robot, in: IEEE International Conference on Systems, Man, and Cybernetics, 2013, pp. 4348–4353.
[6] G.A. Hollinger, S. Singh, A. Kehagias, Improving the efficiency of clearing with multi-agent teams, Int. J. Robot. Res. 29 (8) (2010) 1088–1105.
[7] G. Hollinger, S. Yerramalli, S. Singh, U. Mitra, G. Sukhatme, Distributed data fusion for multirobot search, IEEE Trans. Robot. 31 (1) (2015) 55–66.
[8] J.G. Trafton, A.C. Schultz, D. Perznowski, M.D. Bugajska, W. Adams, N.L. Cassimatis, D.P. Brock, Children and robots learning to play hide and seek, in: Proceeding of the 1st ACM SIGCHI/SIGART Conference on Human-Robot Interaction, 2006, pp. 242–249.
[9] J.R. Anderson, D. Bothell, M.D. Byrne, S. Douglass, C. Lebiere, Y. Qin, An integrated theory of the mind, Psychol. Rev. 111 (4) (2004) 1036–1060.
[10] W.G. Kennedy, M.D. Bugajska, M. Marge, W. Adams, B.R. Fransen, D. Perzanowski, A.C. Schultz, J.G. Trafton, Spatial representation and reasoning for human-robot collaboration, Artif. Intell. (2007) 1554–1559.
[11] E. Martinson, Detecting occluded people for robotic guidance, in: The 23rd IEEE International Symposium on Robot and Human Interactive Communication, 2014, pp. 744–749.
[12] P. Stein, V. Santos, A. Spalanzani, C. Laugier, Navigating in populated environments by following a leader, in: Proceedings of IEEE RO-MAN, 2013, pp. 527–532.
[13] F.-L. Lian, C.-L. Chen, C.-C. Chou, Tracking and following algorithms for mobile robots for service activities in dynamic environments, Int. J. Autom. Smart Technol. 5 (1) (2015) 49–60.
[14] F. Fleuret, J. Berclaz, R. Lengagne, P. Fua, Multicamera people tracking with a probabilistic occupancy map, IEEE Trans. Pattern Anal. Mach. Intell. 30 (2) (2008) 267–282.
[15] G.D. Tipaldi, K.O. Arras, I want my coffee hot! learning to find people under spatio-temporal constraints, in: Proceedings of the IEEE International Conference in Robotics and Automation, ICRA, 2011, pp. 1217–1222.

[16] G. Lidoris, F. Rohrmuller, D. Wollherr, 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, ICRA, 2009, pp. 1416–1422.
[17] P. Trautman, J. Ma, R.M. Murray, A. Krause, Robot navigation in dense human crowds: the case for cooperation, in: IEEE International Conference on Robotics and Automation, ICRA, 2013, pp. 2153–2160.
[18] D. Brscic, T. Kanda, T. Ikeda, T. Miyashita, Person tracking in large public spaces using 3-D range sensors, IEEE Trans. Hum.-Mach. Syst. 43 (6) (2013) 522–534.
[19] M.A. Vzquez, J. Mguez, A robust scheme for distributed particle filtering in wireless sensors networks, Signal Process. 131 (2017) 190–201.
[20] A. Goldhoorn, A. Sanfeliu, R. Alquézar, Comparison of MOMDP and heuristic methods to play hide-and-seek, in: K. Gibert, V.J. Botti, R.R. Bolaño (Eds.), CCIA, in: Frontiers in Artificial Intelligence and Applications, vol. 256, IOS Press, 2013, pp. 31–40.
[21] A. Goldhoorn, R. Alquézar, A. Sanfeliu, Analysis of methods for playing human robot hide-and-seek in a simple real world urban environment, in: ROBOT (2), in: Advances in Intelligent Systems and Computing, vol. 253, Springer, 2013, pp. 505–520.
[22] S.C.W. Ong, S.W. Png, D. Hsu, W.S. Lee, Planning under uncertainty for robotic tasks with mixed observability, Int. J. Robot. Res. 29 (8) (2010) 1053–1068.
[23] A. Amor-Martinez, A. Ruiz, F. Moreno-Noguer, A. Sanfeliu, 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, 2014.
[24] S. Thrun, W. Burgard, D. Fox, Probabilistic Robotics (Intelligent Robotics and Autonomous Agents), The MIT Press, 2005.
[25] M. Montemerlo, S. Thrun, W. Whittaker, Conditional particle filters for simultaneous mobile robot localization and people-tracking, in: Proceedings of the IEEE International Conference on Robotics and Automation, ICRA, vol. 1, IEEE, 2002, pp. 695–701.
[26] A. Sanfeliu, J. Andrade-Cetto, M. Barbosa, R. Bowden, J. Capitán, A. Corominas, A. Gilbert, J. Illingworth, L. Merino, J.M. Mirats, P. Moreno, A. Ollero, J.a. Sequeira, M.T.J. Spaan, Decentralized sensor fusion for ubiquitous networking robotics in urban areas, Sensors 10 (3) (2010) 2274–2314.
[27] K.O. Arras, O.M. Mozos, W. Burgard, Using boosted features for the detection of people in 2D range data, in: Proceedings of the IEEE International Conference on Robotics and Automation, ICRA, 2007, pp. 3402–3407.
[28] S.S. Blackman, Multiple hypothesis tracking for multiple target tracking, IEEE Aerosp. Electron. Syst. Mag. 19 (1) (2004) 5–18.
[29] G. Grisetti, C. Stachniss, W. Burgard, Improved techniques for grid mapping with rao-blackwellized particle filters, J. IEEE Trans. Robot. 23 (1) (2007) 34–46.
[30] M. Arulampalam, S. Maskell, N. Gordon, T. Clapp, A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking, IEEE Trans. Signal Process. 50 (2) (2002) 174–188.
[31] B.P. Gerkey, K. Konolige, Planning and control in unstructured terrain, in: In Workshop on Path Planning on Costmaps, Proceedings of the IEEE International Conference on Robotics and Automation, ICRA, 2008.
[32] G. Ferrer, A. Garrell, A. Sanfeliu, Robot companion: A social-force based approach with human awareness-navigation in crowded environments, in: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, 2013, pp. 1688–1694.

Alex Goldhoorn is a Ph.D. student in Automatic, control and vision at the Universitat Politècnica de Catalunya, Barcelona, Spain. He obtained his M.Sc. degree in Artificial Intelligence from the University of Groningen (NL) in 2008, and he obtained his B.Sc. degree in Computer Science from the University of Groningen in 2006. His main research interests are machine learning and mobile robotics.

Anaís Garrell Zulueta received the BSEE degree of Mathematics from Universitat de Barcelona (UB), Spain, in 2006, and the Ph.D. degree in Automatic, Control and Vision from the Universitat Politècnica de Catalunya, Barcelona, Spain, in 2013, and obtained the second prize of the best Ph.D. Award in robotics in Spain. She has participated on diverse national and European projects and has worked with other research groups including the Robotics Institute of CMU (US). Her research interests are focused on machine learning, mobile robotics and human–robot interaction.

René Alquézar obtained the Ph.D. degree in Informatics at the Universitat Politècnica de Catalunya (UPC) in 1997. He was technical manager of R+D projects at the Spanish company NTE (*Nuevas Tecnologas Espaciales*), S.A. from 1987 to 1991 and held a pre-doctoral grant at the *Institut de Cibernètica* (CSIC-UPC) from 1991 to 1994. Since 1994 he has been at the Computer Science department of UPC, first as assistant professor and since 2001 as associate professor. He has been researcher at the *Institut de Robòtica i Informàtica Industrial* (IRI, CSIC-UPC) since 2005 and member of the research group in Mobile Robotics and Intelligent Systems. He is member of the steering committee of the Catalan Association for Artificial Intelligence since 2012 and vice-dean head of graduate studies of the Barcelona School of Informatics (FIB-UPC) since 2013. His research areas of interest include: neural networks, machine learning, data mining, pattern recognition, computer vision and mobile robotics.

Alberto Sanfeliu received the BSEE and Ph.D. degrees from the Universitat Politcnica de Catalunya (UPC), Spain, in 1978 and 1982, respectively. He joined the faculty of UPC in 1981 and is full professor of Computational Sciences and Artificial Intelligence. He has been the director of the Institut de Robòtica i Informàtica Industrial (UPC-CSIC), past director of the Automatic Control Department of UPC, director of the Artificial Vision and Intelligent System Group (VIS) and past president of AERFAI, and he is doing research at Institut de Robòtica i Informàtica Industrial—IRI, (UPC-CSIC). He has worked on various theoretical aspects on pattern recognition, computer vision and robotics and on applications on vision defect detection, tracking, object recognition, robot vision and SLAM. He has several patents on quality control based on computer vision. He has authored books in pattern recognition and SLAM, and published more than 200 papers in international journals and conferences. He has lead and participated in 30 R&D projects, 8 of them funded by the European Commission, and he is currently coordinator of the European project URUS (Ubiquitous Networking Robotics in Urban Areas). He is (or has been) member of editorial boards of Computer Vision and Image Understanding, International Journal on Pattern Recognition and Artificial Intelligence, Pattern Recognition Letters, Computacin y Sistemas and Electronic Letters on Computer Vision. He received the prize to the Technology given by the Generalitat de Catalonia and is Fellow of the International Association for Pattern Recognition. He is member of IEEE.