Searching and Tracking People in Urban Environments with Static and Dynamic Obstacles

Alex Goldhoorn[a,*], Anaís Garrell[a], René Alquézar[a], Alberto Sanfeliu[a]

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

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.

Keywords: Urban Robotics, Search and Rescue Robots, Human-robot-interaction, Particle Filter

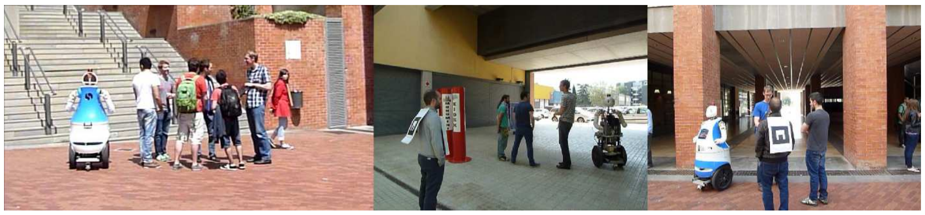


Figure 1: Dabo performs the search-and-track task with a target (wearing a tag for recognition) in different urban environments.

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 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

*Corresponding author
Email address: agoldhoorn@iri.upc.edu (Alex Goldhoorn)

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 5.1.

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 **??** In Section 4, the experimental setup is introduced and in Section 5 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.

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.

3. 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.

3.1. 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.

3.2. Modifications of the Basic Particle Filer 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

Algorithm 1 A basic particle filter.

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

Algorithm 2 Initialization of the modified particle filter.

```
1: function INIT($o$,$N$)
2:     $S = \emptyset$
3:     for $i = 1$ to $N$ do
4:         if $o_{\text{person}} = \emptyset$ then                ▷ i.e. person not visible
5:             $s$ = RANDOMNOTVISIBLEPOS(o)
6:         else
7:             $s = o$
8:         end if
9:         $s = \mathcal{N}(s, \sigma_{\text{person}} I)$
10:        $S = S \cup \{s\}$
11:    end for
12:    return $S$
13: end function
```

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):

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

The update step was modified, see Algorithm 3, such that the particle weight

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:     $\forall_{s \in S} : s_w = w(s,o)$
3:     $\forall_{s \in S} : s_w = s_w / \sum_{k \in S} k_w$                ▷ normalize
4:     $\bar{S} = \emptyset$
5:     for $i = 1$ to $N$ do
6:         $\bar{s}$ sample from $S$                          ▷ with probability $\bar{s}_w$
7:         $\bar{S} = \bar{S} \cup \{\bar{s}\}$
8:     end for
9:     $S = \bar{S}$
10: end function
```

is changed, even if no observation is available of the person:

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

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 can not 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 $w_{\text{inc}} << w_{\text{cons}}$ is used (we set it to 0.001). The visibility probability is defined as the following piecewise linear formula:

$$P_{\text{vis}}(o,s) = \begin{cases} 0, & \text{if } \textsc{RayTrace}(o_{\text{agent}}, s) \text{ not free} \\ p_{\text{v,max}}, & \text{if } d < d_{\text{v,max}} \\ \max(0, p_{\text{v,max}} - \alpha_{\text{vis}}(d - d_{\text{v,max}})), & \text{otherwise} \end{cases} \tag{3}$$

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 $\alpha_{\text{vis}} = 0.17$.

3.3. 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.

3.4. 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.

4. 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.

4.1. 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.

4.2. People Recognition And Dynamic Obstacles

4.3. Robot Mapping and Navigation

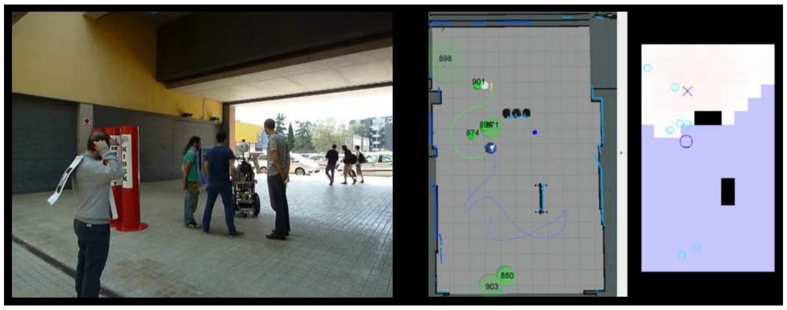
4.4. Environments and Maps

Since there are no standard search-and-track data sets, we have used a large environment (*Telecos Square* Fig. 2(b) and 2(c), 60 m $\times$ 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 $\times$ 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.

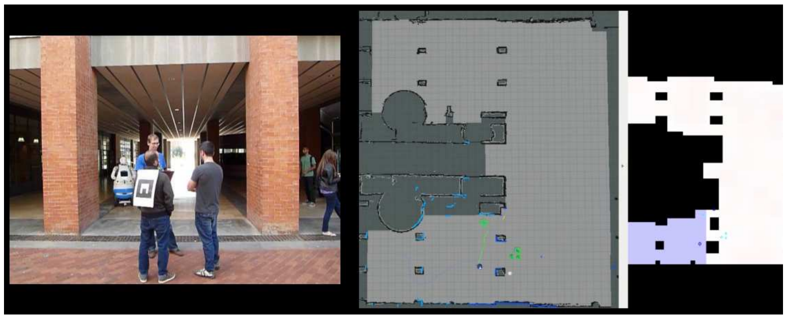
The *FME* environment is outdoor, but partly covered by a roof, and since no obstacles were in the field we placed several artificial ones, as can be seen in Fig. 2(a). The *Telecos Square* is the largest environment which contains a square, and two covered areas with several columns. Through both areas people pass by frequently, especially the last since it is the center of the campus.

5. Simulations and Real-life Experiments

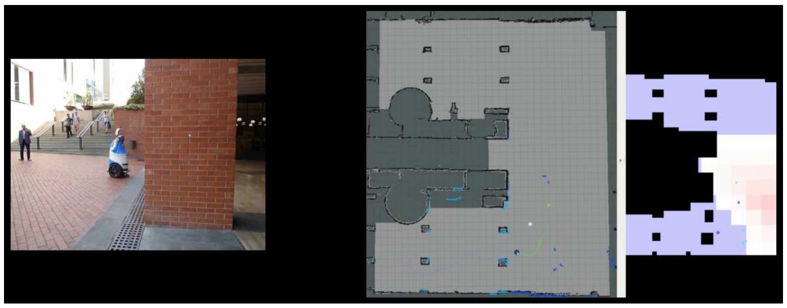
To verify the methods, first we did simulations, and thereafter real-world experiments were done in urban environments.



(a)



(b)



(c)

Figure 2: 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.

5.1. Simulation

The maps contain discrete cells which either are free or obstacles, . 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 3. 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.

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.

5.2. 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.

5.3. Results

The results of the simulations, the 140 repetitions, are shown in

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

Table 1: The parameter values used during the real experiments and simulations. *For simulation we use discrete time steps, for the real experiments seconds are used.

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 $\times$ m)	1×1	3.8×3.8
Belief update time*	3 s; 3 steps	
$d_{\text{max\_search}}$ (m)	10	25

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. 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.

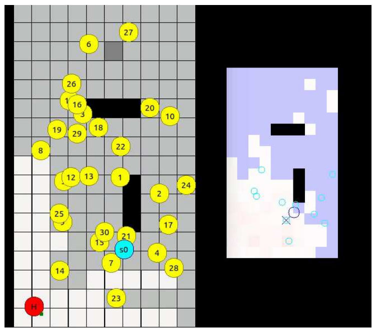
Using the dynamic obstacles, which have been detected to update the probability map, has a great advantage in certain situations, as shown in Fig. 3.

Table 2:

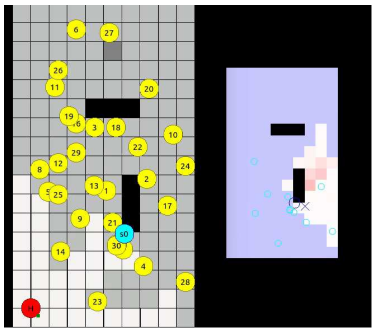
Type of Measurement	Num. Pers.	*See All F.*	*HB-PF S&T*(d)	*HB-PF S&T*	*Ad.HB-CR-POMCP F.*(d)	*Ad.HB-CR-POMCP F.*
First visible step	0	2.4 ± 3.2	4.5 ± 5.8	4.7 ± 8	5.3 ± 6.9	5 ± 6.5
(time steps)	10	3.1 ± 3.7	7.2 ± 9.1	6.9 ± 10.7	7.7 ± 11.8	7.4 ± 10.1
(search)	100	8.4 ± 6.4	67.4 ± 71.1	90 ± 111.1	65.2 ± 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

Table 3:

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



(a)



(b)

Figure 3: (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.

Left can be seen the situation where there are particles maintained behind the detected dynamic obstacles, whereas in Fig. 3(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/`.

5.4. Real-life Experiments

6. Conclusion

Acknowledgements

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] 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, Proceeding of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction (2006) 242–249.

[6] J. R. Anderson, D. Bothell, M. D. Byrne, S. Douglass, C. Lebiere, Y. Qin, An integrated theory of the mind., Psychological review 111 (4) (2004) 1036–60.

[7] 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, Artificial Intelligence (2007) 1554–1559.

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

[9] 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.

[10] 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), Vol. 253 of Advances in Intelligent Systems and Computing, Springer, 2013, pp. 505–520.

[11] 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.