UN NUEVO MÉTODO COOPERATIVO PARA ENCONTRAR PERSONAS EN UN ENTORNO URBANO CON ROBOTS MÓVILES

Alex Goldhoorn, Anaís Garrell, René Alquézar y Alberto Sanfeliu
Institut de Robòtica i Informàtica Industrial (CSIC-UPC). Llorens Artigas 4-6, 08028 Barcelona.
{agoldhoorn,agarrell,ralqueza,sanfeliu} @iri.upc.edu

Resumen

Se presenta un nuevo método para localizar a personas en entornos urbanos usando robots móviles sociales que trabajan de manera cooperativa, el cual supera las limitaciones de enfoques ya existentes, que se adaptan a entornos específicos, o se basan en comportamientos humanos poco realistas. Con este método cooperativo los robots pueden encontrar a personas fuera del campo de rango de sensores u ocultados por obstáculos dinámicos o estáticos. Nuestro enfoque incluye la búsqueda de personas, seguimiento, cooperación multi-robot y comunicación. En particular se define un "Cooperative Highest-Belief Continuous Real-time POMCP" que puede ejecutarse en tiempo real y en entornos continuos y grandes. En este método se usan algoritmos de búsqueda online Partially Observable Monte-Carlo Planning *(POMCP), los cuales, al contrario de trabajos anteriores son capaces de planificar con incertidumbre y con grandes espacios de estados. La estrategia de búsqueda hace un balanceo entre la probabilidad de que la persona esté en una posición concreta, la distancia de las posiciones, y si la posición está cerca de una meta ya asignada a otro robot. Se ha validado el método con extensivo número de simulaciones y experimentos reales con una persona y dos robots.*

Palabras clave: Robótica móvil, Robótica urbana, Interacción hombre-robot.

1 INTRODUCCIÓN

En los últimos años se ha incrementado el interés en robots de servicio en entornos urbanos. Hay una gran cantidad de aplicaciones, desde la exploración automática [18] hasta el transporte y evacuación de personas en situaciones de emergencia [3]. También se está trabajando en robots que pueden operar como miembros de un equipo [12]. En concreto, para acompañar y servir a humanos en áreas grandes, los robots autónomos deberían poder localizarse y navegar con personas de una manera segura y natural [6].

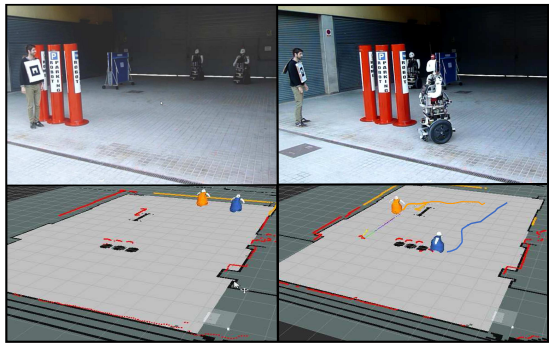

Figura 1: Los robots buscan cooperativamente al peatón explorando diferentes partes del entorno y compartiendo sus observaciones y objetivos.

En esta investigación se ha usado un equipo de robots cooperativos para localizar y seguir a personas en entornos urbanos. Las personas podían esconderse y los robots, usando el mapa, tenían que encontrarlas y seguirlas, por ello, se presenta un método nuevo, el *Cooperative Highest-Belief Continuous Real-time POMCP* (Cooperative HB-CR-POMCP).

Anteriormente [8] se han presentado y evaluado diferentes métodos para encontrar y buscar a personas en entornos urbanos con un robot de servicio. Estos métodos usaban el algoritmo de planificación POMCP (*Partially Observable Monte-Carlo Planning*), los cuales, al contrario que otros enfoques, pueden planificar en entornos grandes, y en situaciones de incertidumbre. Además, la planificación se ejecuta en tiempo real y de manera continuo. En este trabajo, se va más allá usando un equipo de dos robots cooperativos que trabajan juntos en un entorno dinámico y que comparten información para mejorar el rendimiento de la tarea. En la estrategia de búsqueda se tiene en cuenta la probabilidad que la persona esté en un sitio concreto, la distancia a aquella posición y si el objetivo está cerca de la meta del otro robot.

El método nuevo, además tiene en cuenta el ruido de los sensores y la detección de falsos positivos y negativos. En caso de fallo de comunicación, por ejemplo, también son capaces de ejecutar la tarea solos.

Finalmente, la validación del modelo se ha realizado a través de un número extensivo de experimentos y simulaciones, véase la figura 1. Se demuestra que usando robots cooperativos mejora el rendimiento de la tarea. Para los experimentos reales se han usado los robots móviles sociales Tibi y Dabo [5].

En el resto del artículo se introducen trabajos relacionados de búsqueda con robots cooperativos, después se explica el modelo en sección 3. Los detalles de las simulaciones y experimentos reales se muestran en sección 4 y se finaliza con las conclusiones.

2 TRABAJOS ANTERIORES

En las últimas décadas se han realizado muchos trabajos en el campo de robots móviles autónomos para entornos exteriores. Se han realizado sistemas de navegación robustos [14], evitación de obstáculos, y localización [4], los cuales se han integrado en diferentes plataformas robóticas [5]. Un número de métodos se han desarrollado para permitir que los robots naveguen de manera segura cerca de personas [21].

Concretamente, el problema de exploración de entornos urbanos con un grupo de robots móviles es altamente complejo y se ha estudiado. Por ejemplo, Yamauchi et al. [20] presentaron una técnica para aprender mapas con un grupo de robots. Además, introdujeron la idea de una frontera que separa el entorno en partes conocidas y desconocidas. En [2] se presenta una técnica para coordinar equipos de robots móviles. Otros [7] propusieron una arquitectura de exploración con equipos de robots móviles haciendo subastas para dividir tareas entre robots. En estos trabajos normalmente se asume que las conexiones de red tengan suficiente ancho de banda.

Hay muchos trabajos sobre la búsqueda de personas en interiores, como en [19] donde un robot de servicio detecta y busca personas dentro de una casa mirando a las piernas, la cara y la forma del cuerpo y movimiento de la persona. Sin embargo, su método no se puede usar en el aire libre y se supone que la persona está en la misma posición durante el experimento. Johanson et al. [11] presentaron una simplificación del juego de escondite, donde hay un robot buscando a uno o más jugadores que se esconden. Este juego también requiere un gran número de funciones cognitivas como por ejemplo: buscar y navegar, coordinación de la búsqueda, anticipación y planificación.

En [10] y [9] se enfocó al juego del escondite, porque es más limitado y da facilidad en la investigación del problema de la búsqueda de personas. Primero se empezó a trabajar en espacios discretos [9] usando MOMDPs (Mixed Observable Markovian Decision Processes, [15]) para buscar personas. En [8] se extendió el POMCP [17] para poder usarlo en tiempo real y en espacio continuo (CR-POMCP, [8]). En este artículo se extiende el trabajo anterior usando dos robots cooperativos que comunican observaciones y planes para encontrar a la persona más rápida y eficiente.

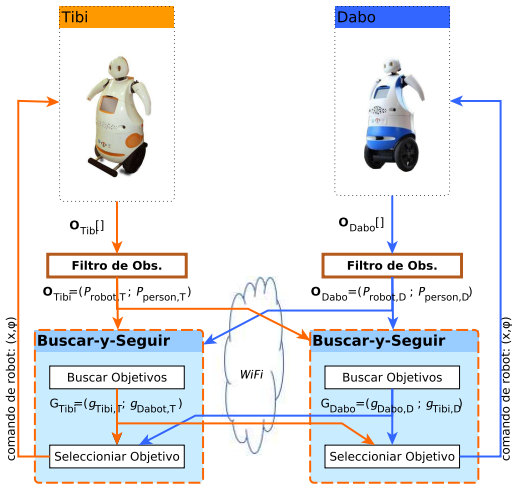

Figura 2: Los robots usan *Cooperative POMCP* para buscar y seguir a una persona cooperativamente compartiendo observaciones y planes usando una red inalámbrica. Los robots generan una lista de observaciones $O[]$, las cuales se filtran y resultan en una observación para el robot. Las letras D y T en el subíndice de P y g se refieren a quien ha generado el objetivo (gol, G) o la observación.

3 Cooperative HB-CR-POMCP

Se presenta un método para buscar y seguir personas con un equipo de robots, véase figura 2. En esta sección se explican los dos componentes del procedimiento: *(i)* Adaptive Highest Belief CR-POMCP Follower y la nueva *(ii)* Coordinación del equipo de robots.

3.1 Adaptive HB-CR-POMCP Follower

En esta sección se explica el algoritmo *Adaptive Highest Belief Continuous Real-Time POMCP Follower*, previamente presentado en [8], para buscar y seguir una persona con un robot. El método utiliza *Partially Observable Monte-Carlo Planning* (POMCP) [17], que al contrario a otros enfoques, puede planificar con incertidumbre y en espacios de estado largos. POMCP estima el estado actual y en este trabajo la localización de la persona, usando las observaciones con lo cual propone

la mejor acción para el agente. El POMCP fue adaptado a *Continuous Real-time POMCP* (CR-POMCP) [8] en el que los estados son continuos. Este método funciona en tiempo real, en estados continuos y con obstáculos dinámicos (por ejemplo otras personas paseando).

El método POMCP está basado en "Partially Observable Markovian Decision Processes" (POMDPs, [16, 13]), pero en vez de calcular la recompensa esperada o función de valor (*value function*) usando *value iteration* [16] se usan simulaciones de Monte Carlo para estimar la función de valor. Los modelos POMDP tienen estados S, observaciones O, acciones A, recompensas R, la probabilidad de cambiar de estado en una acción ($P(s'|s,a)$) y la probabilidad de hacer una observación ($P(o|s',a)$). El POMCP no necesita las funciones de probabilidad pero usa un simulador POMDP $(s',o,r) = \mathcal{G}(s,a)$ el cual genera el siguiente estado s', la observación o, y recompensa r para estado actual s y la acción realizada a. En vez de conocer el estado actual, los dos modelos mantienen una creencia (*belief*), la cual, en el POMDP, es la probabilidad de estar en cada posible estado, y en el POMCP es una lista de posibles estados. Los estados son una combinación de la posición del robot y la persona: $(s_{\text{robot}}, s_{\text{person}})$, y las observaciones son iguales, pero la observación de la persona puede ser *oculta*. Además, se definen nueve acciones de movimiento para el robot: estar quieto o en movimiento hacia una de las ocho direcciones equidistantes. Se dan recompensas en base del estado actual, en este trabajo es la distancia del camino más corto en negativo entre el robot y la persona ($-\|r-p\|$). Se inicializa el sistema con un *belief* b_0, el cual se basa en la observación inicial o_0 del robot. Inicialmente b_0 se distribuye entre las localizaciones no visibles si la persona no es visible, si no se concentra cerca de la observación. Para comprobar la visibilidad se usa un algoritmo de trazados de rayo ya que se conoce el mapa del entorno de antemano.

A continuación, se genera un árbol haciendo un gran número de simulaciones usando el simulador de POMDP $\mathcal{G}$. En este método de búsqueda y seguimiento se simula el movimiento del robot haciendo su acción y las personas se mueven aleatoriamente. Para hacerlo más realista se añade ruido a s' y o, y se simulan falsos positivos y negativos.

El árbol creado [8, 17] consiste en *nodos de belief* y *nodos de acción*. Cada nodo mantiene un registro de la recompensa esperada V y el número de veces N que una simulación ha pasado por aquel nodo. La raíz es un *nodo de belief* la cual contiene el *belief* actual y, durante la simulación, el *belief* de segundo nivel del árbol crece porque se añaden

Algoritmo 1 El planificador POMCP. Recuperando nodos hijos se escribe como $Node[a]$ (en el caso de acción a por ejemplo); d es la profundidad.

```
1: function SimNode(Node,s,d)
2:     if d > d_max then return 0
3:     else
4:         a ← argmax_a Node[a].V + c sqrt(log(Node.N) / Node[a].N)
5:         if d = 1 then Node.B = Node.B ∪ {s}
6:         (s', o, r_immediate) ← G(s, a)
7:         if not Node[a][o] exists and Node[a].N ≥ e_count then
8:             Add Node[a][o]
9:         end if
10:        if Node[a][o] exists then
11:            r_delayed ← SimNode(Node[a][o],s',d+1)
12:        else
13:            r_delayed ← Rollout(s',d+1)
14:        end if
15:        r_total ← r_immediate + γ r_delayed
16:        Node[a].N ← Node[a].N + 1
17:        Node[a].V ← Node[a].V + (r_total − Node[a].V) / Node[a].N
18:        Node.N ← Node.N + 1
19:        Node.V ← Node.V + (r_total − Node.V) / Node.N
20:        return r_total
21:    end if
22: end function
```

los estados resultantes s' (algoritmo 1, linea 5). Cada *nodo de belief* tiene un nodo de acción por cada posible acción y después de haber hecho la observación se llega a otro *nodo de belief*.

El algoritmo 1 muestra como se ejecuta la simulación hasta llegar a un máximo de profundidad $d_{\max}$. Se elije una acción encontrando el nodo con el máximo valor V y para explorar otras acciones se ha añadido el término de la derecha en la linea 4, donde c es la constante de exploración. El valor de esta constante se ha definido como $c = r_{\text{hi}} - r_{\text{lo}}$ (la diferencia entre la recompensa más alta r_{hi} y la más baja r_{lo} [17]). El número de expansión e_{count} (linea 7) se usa para prevenir que el árbol crezca demasiado rápido. Durante la simulación se atraviesa el árbol y cuando un *nodo de belief* todavía no existe (para ciertas acciones), la simulación sigue una política *rollout*, la cual es un movimiento aleatorio hasta llegar a una profundidad máxima. La recompensa total se calcula en la linea 15 basada en la recompensa instantánea (para el estado) y la recompensa obtenido por hacer los siguientes pasos, el último se pondera con un factor de descuento γ.

Cuando se ha generado el árbol, se puede elegir la mejor acción, es decir, el nodo de acción con la recompensa esperada V más alta. Después de haber ejecutado la acción a y de haber recuperado su observación o, se elige una nueva raíz para el árbol. La raíz nueva es el nodo que se obtenga siguiendo desde la raíz la acción y observación. La raíz nueva ya tendrá algunos puntos de *belief* de

simulaciones anteriores y se añaden puntos nuevos hasta llegar los n_{belief} estados. La generación de estados de *belief* nuevos se hace eligiendo aleatoriamente estados $s \in Root_{\text{old}}.\mathcal{B}$ del *belief* anterior y aplicando el simulador $(s', o, r) = \mathcal{G}(s, a)$ para obtener nuevos estados s', y añadirlos al $Root.\mathcal{B}$ si son consistentes con la observación o.

La complejidad de la mayoría de los algoritmos de tipo POMDP es exponencial en el número de acciones y observaciones, el algoritmo POMCP depende solo de la complejidad del simulador de POMDP y es lineal con el número de simulaciones n_{sim}.

Para los experimentos reales con el robot Dabo [5] se han hecho varias adaptaciones, las cuales se usaron en el *Adaptive Highest Belief CR-POMCP Follower*. Haciendo experimentos reales con el algoritmo CR-POMCP se ha demostrado que los robots se mueven lentamente y por eso no eran capaces de encontrar y seguir a la persona en tiempo real. Estos problemas ocurrieron porque: 1) los pasos de cada acción eran pequeños; 2) había un número de direcciones limitadas; y 3) las acciones cambiaban demasiados rápido. Las primeras dos limitaciones fueron causadas por el método de planificación porque el número de acciones se incrementa a causa un crecimiento exponencial en el árbol de búsqueda. El tercer problema se produce por el ruido durante el aprendizaje de la política, que provienen de las simulaciones de Monte Carlo, y unas diferencias pequeñas de acciones como *norte* y *nordeste*.

Una manera de solucionar este problema es permitir al robot ir más lejos y no solo un paso. Se propuso el *Highest Belief CR-POMCP* (HB-CR-POMCP) para coger como objetivo el punto con el *belief* más alto, y después se usa el planificador de caminos para llegar. Se actualizan las metas cada $t_{\text{hb.update}}$ segundos, o cuando esté visible la persona. Se calcula el *belief* más alto (*highest belief*) generando primero una matriz histograma, y la posición del *belief* más alto es elegido. En figura 3 se muestran varios mapas y *belief* (la fila más abajo), donde los cuadros blancos a rojos indican una probabilidad de bajo a alto, y azul claro representa una probabilidad de 0. Hay dos desventajas con este método: primero, la elección de una resolución de la matriz es difícil, porque una resolución demasiado baja reduce la precisión, y una resolución demasiado alto genera más ruido; segundo, el método no tiene en cuenta la distancia al objetivo.

Se resolvió el primer problema combinando el HB-CR-POMCP con el *Heuristic Follower* [8], el cual directamente va a la persona si es visible. El segundo problema se ha resuelto limitando la distancia de búsqueda para una meta. Se explican más detalles del algoritmo en [8].

Algoritmo 2 El planificador para dos agentes, donde cada uno de los agentes ejecuta este algoritmo. Las observaciones o_1 y o_2 son de robot 1 y 2 respectivamente; g_1^1, g_2^1 son las metas de los robots y b_i^j es la probabilidad que está la persona en el goal g_i^j.

1: $O[] \leftarrow$ RetrieveObservations( )
2: $o_1 \leftarrow$ FilterObservation(O)
3: SendObservation(o_1)
4: $o_2 \leftarrow$ ReceiveObservation( )
5: $\langle g_1^1, g_2^1, b_1^1, b_2^1 \rangle \leftarrow$ FindGoals(o_1,o_2)
6: SendGoals($\langle g_1^1, g_2^1, b_1^1, b_2^1 \rangle$)
7: $\langle g_1^2, g_2^2, b_1^2, b_2^2 \rangle \leftarrow$ ReceiveGoals( )
8: $p \leftarrow$ SelectGoal($\langle g_1^1, g_2^1, b_1^1, b_2^1 \rangle$, $\langle g_1^2, g_2^2, b_1^2, b_2^2 \rangle$)

3.2 Coordinación de un equipo de robots

La búsqueda de una persona tendría que ser más rápida con dos agentes, o como mucho igual de rápido como con un agente, si los agentes no se interfieren. Sin embargo, la búsqueda robusta y cooperativa no es evidente. Se propone un método que usa el método explicado previamente: HB-CR-POMCP, aplicado al uso de dos agentes. Se comparte las observaciones y se decide adonde tiene que ir cada robot para buscar la persona usando un método de exploración basado en [2]. En caso de fallo o retraso de comunicación solo usan las observaciones propias.

Los estados usados en el modelo de multi robot son los mismos definidos previamente, porque añadir la posición del otro robot al estado y la observación causaría que creciera el número de estados exponencialmente. Igualmente, se usa la observación del otro agente para poder filtrar el *belief*.

Cada robot ejecuta el mismo algoritmo (algoritmo 2, figura 2), compartiendo sus observaciones. Ambos robots también calculan la meta de cada uno si han recibido información del otro. En la notación, el subíndice y superíndice se use 1 para indicar que es del mismo robot y 2 para el otro robot. El superíndice de las metas (*goals*) y *beliefs* indican el robot que realiza el cálculo.

En el método (algoritmo 2), primero se obtiene una lista de las observaciones de la persona detectada. La observación filtrada se envía al otro robot (o_2). Estas observaciones se usan para actualizar el *belief*, pero primero se comprueba la consistencia para decidir cuales de las observaciones se usan. Se considera que son consistes si la persona solo es visible para un robot, o si las dos observaciones de la persona están cercanas. Am-

Algoritmo 3 El explorador busca las metas g_1 y g_2 para los dos robots usando (1).

1: $T \leftarrow$ GetHighestBeliefPoints($\mathcal{B}$,n_{hb})
2: $\forall_{t \in T} U_t = 1$
3: $g_1 \leftarrow \arg\max_{t \in T}\ s_{\text{EXPL}}(s_1,t)$
4: $\forall_{t \in T} U_t = U_t - P(\|t - g_1\|)$
5: $g_2 \leftarrow \arg\max_{t \in T}\ s_{\text{EXPL}}(s_2,t)$

bos robots actualizan sus *beliefs* con la posición de la persona si era visible y si las observaciones eran consistentes, si no se usan las dos observaciones o_1 y o_2 en la generación de los estados $s' \in \mathcal{B}$ para el *belief*, los cuales eran consistentes con o_1 con una probabilidad de p_{own} o consistentes con o_2 en el otro caso. En caso de no haber recibido la otra observación, se usa solo la del propio robot.

El *belief* nuevo se coge del árbol siguiendo las acciónes y observaciones obtenidas. Antes que el árbol empiece a crecer se filtran los estados de la raíz nueva para que tenga solo estados consistentes con las observaciones. Para manejar los falsos positivos, por ruido del sensor u obstrucción temporal (por otra persona por ejemplo), se permiten estados inconsistentes en el *belief* con una probabilidad de p_{fn}.

Después de haber actualizado el *belief* se calculan los objetivos para cada robot. Si la persona era visible y las observaciones eran consistentes, entonces siguen a la persona de lado a lado [6]. De otra manera, los robots exploran las áreas donde haya más probabilidad. En vez de solo elegir un único punto (*highest belief*), se calculan los n_{hb} puntos con más *belief*.

El algoritmo 3 muestra la exploración del espacio de *belief* y se base en el explorador de Burgard et al. [2]. Cada robot calcula la meta de cada robot. Para cada punto de *belief* más alto $t \in T$ se calcula una puntiación para cada agente a:

$$s_{\text{expl}}(a,t) = w_u U_t + w_d \frac{-\|a-t\|}{d_{\max}} + w_b \frac{b_t}{b_{\max}} \quad (1)$$

donde U_t es la función de utilidad para la posición t, y b_t representa el *belief* en t. El segundo y tercer término son normalizados con la distancia máxima $d_{\max}$, y el *belief* $b_{\max}$ con respeto a la lista de metas potenciales T. La utilidad U_t [2] se inicializa con 1 (linea 2 de algoritmo 3), y se actualiza antes de calcular la meta del otro robot (linea 4), con g_1 siendo un objetivo ya asignado, y:

$$P(d) = \begin{cases} 1.0 - \frac{d}{r}, & \text{if } d < r \\ 0, & \text{de otro modo} \end{cases} \quad (2)$$

donde r es el rango dentro de cual se reduce la probabilidad de elegir una meta para otro robot.

Tabla 1: Selección de los objetivos basado en la distancia entre en la meta asignada, donde $G_i = (g_1^i, g_2^i)$ y $B_i = (b_1^i, b_2^i)$.

		$\|g_1^1 - g_1^2\|$ **pequeño**	$\|g_1^1 - g_1^2\|$ **grande**
$\|g_2^1 - g_2^2\|$	**peq.**	G_1, if $max(B_1) > max(B_2)$ G_2, de otro modo	G_1, if $b_1^1 > b_1^2$ G_2 de otro modo
$\|g_2^1 - g_2^2\|$	**gran.**	G_2 if $b_2^2 > b_2^1$ G_1, de otro modo	G_1, if $\max(B_1) > \max(B_2)$ G_2, de otro modo

Los términos de (1) se balancean con w_u, w_d, y w_b, cuales se suman a 1.

Los dos robots calculan los objetivos de ambos robots, usando el método de exploración, o como seguidor. Se minimiza la suma de las distancias ($\|s_i - g_j\|$) para asegurar que los dos robots elijan la meta más cercana. Estas metas se envían al otro robot (algoritmo 2) y entonces cada robot elige su meta basado en su semejanza y el *belief* de la meta. Se usa la distancia entre los objetivos para decidir cual se elige, como se lista en tabla 1. Como en el método anterior, para prevenir que se actualice demasiado a menudo el objetivo, se mantiene la meta durante un tiempo t_{update} si la persona no es visible.

4 SIMULACIONES Y EXPERIMENTOS

En esta sección se explica la configuración de los experimentos, a continuación, se introducen los resultados.

4.1 CONFIGURACIÓN EXPERIMENTAL

Los experimentos se han hecho con dos robots móviles, Tibi y Dabo, en un entorno urbano (figura 3).

4.1.1 ROBOT PLATAFORMA

Los robots fueron diseñados para trabajar en zonas peatonales e interactuar con personas, tienen una plataforma de dos ruedas Segway RMP200, y tienen codificadores de ruedas que proporcionan el odómetro. Además, los robots pueden interactuar con las personas a través de una pantalla táctil, altavoces, expresiones de cara (con LEDs) y brazos y una cabeza movibles. Los robots usan WiFi para comunicarse entre ellos.

El reconocimiento de personas (a través de reconocimiento de cara por ejemplo) no es el enfoque de esta investigación, por lo cual se han usado

marcadores AR (Augmented Reality Markers; [1]) para detectar personas. La persona a quien se busca tiene que llevar el marcador como se muestra en figura 3. Para filtrar los falsos positivos de la detección del marcador se usa también la detección de piernas con láser y se aceptan solamente las detecciones del marcador que están cerca de una detección de piernas. Como efecto secundario, ocurren falsos negativos, pero estos pueden ser tratados por el algoritmo *CR-POMCP*.

Para navegar hacia las metas se han utilizado los algoritmos de ROS. Consisten en un planificador global de Dijkstra, el cual usa un mapa pregenerado y un planificador local (*Trajectory Roll Out planner*) que genera y puntúa trayectorias sobre un mapa de coste que se actualiza con datos del láser de rango.

4.1.2 ENTORNOS

Los mapas de los entornos son discretos, pero las coordenadas usadas por los agentes son continuas. Los células de la cuadrícula pueden ser o *libres* o un *obstáculo*. Los agentes no pueden estar ni fuera del mapa ni dentro de un obstáculo. Se han hecho simulaciones con dos mapas urbanas: el laboratorio de Robots de Barcelona (80 m × 15 m), y FME (Facultat de Matemàtiques i Estadística) lab (17 m × 12 m).

4.2 SIMULACIONES

En las simulaciones, cada persona solo puede hacer como máximo un paso. Ya que los algoritmos pueden dar metas más lejos que un paso, se usa el camino más corto para llegar a ellas. Las personas simuladas empiezan en una posición aleatoria y con una meta aleatoria. En cada iteración, la persona hace un paso hacía la meta y cuando llegue a la meta cambia a otra meta aleatoria. Se ha generado un entorno de multi-personas añadiendo un grupo de diez personas que caminan por el escenario que ocultan de vez en cuando la persona buscada.

En las simulaciones, se han comparado el nuevo *Cooperative HB-CR-POMCP* con el *Adaptive HB-CR-POMCP* – aquí usando solo un robot – y *Independent HB-CR-POMCP*, donde dos robots independientes hacen la tarea sin compartir información. Se ha medido la distancia media hacia la persona y el porcentaje del tiempo que el robot veía a la persona. Se han usado los valores de parámetros definidos en [8], y los diferentes parámetros para el *Cooperative HB-CR-POMCP*, se han obtenido experimentalmente y se muestra en tabla 2. Como los pesos de (1) se han elegido los mejores: $w_u = 0.4$, $w_d = 0.4$ y $w_b = 0.2$.

Para hacer la comparación lo más justa posible, la posición de inicio del robot y el camino de la persona son los mismos para cada método. Se han

Tabla 2: Los valores de los parámetros usados en simulación y experimentos reales.

Param.	**Real**	**Sim.**	**Param.**	**Real**	**Sim.**
r	10	25	$d_{\max}$	1	1
p_{own}	0.6	0.6	d_{small}	4	4
t_{update}	3 s	3 steps	n_{hb}	5	10

Tabla 3: Resultados de las simulaciones con los 3 métodos: el promedio ± error estándar de la **dist**ancia a la persona y la **v**isibilidad, el tiempo (% del total) que la persona ha sido visible. La primera columna indica el mapa y el número de personas paseando en la simulación como obstáculos dinámicos.

		MÉTODO	**Dist. [m]**	V. [%]
FME LAB	0 Pers.	Ad. HB-CR-P. Fol.	1.33 (±0.13)	97
		Independent HB-CR-P.	1.15 (±0.08)	99
		Cooperative HB-CR-P.	1.14 (±0.07)	99
	10 Pers.	Ad. HB-CR-P. Fol.	2.70 (±0.23)	89
		Independent HB-CR-P.	1.81 (±0.13)	95
		Cooperative HB-CR-P.	1.37 (±0.08)	98
BRL LAB	0 Pers.	Ad. HB-CR-P. Fol.	3.64 (±0.78)	95
		Independent HB-CR-P.	2.42 (±0.54)	98
		Cooperative HB-CR-P.	2.23 (±0.49)	98
	10 Pers.	Ad. HB-CR-P. Fol.	7.54 (±1.13)	86
		Independent HB-CR-P.	3.73 (±0.68)	95
		Cooperative HB-CR-P.	2.64 (±0.50)	97

hecho más que 2.000 simulaciones repitiendo cada condición por lo menos 100 veces.

La tabla 3 muestra la distancia promedia entre los robots y la persona buscada. Se puede ver que en los diferentes entornos y abajo diferentes condiciones el *Cooperative HB-CR-POMCP* funcione mucho mejor que los otros métodos. Además se muestra la visibilidad (**V.**) de los robots, es decir, el porcentaje uno de los agentes ha detectado la persona.

Finalmente, para la evaluación se ha usado la prueba de Wilcoxon (bilaterales). En conclusión el *Cooperative HB-CR-POMCP* funcione mejor que *Adaptive HB-CR-POMCP Follower* ($p < 0.01$) y *Independent HB-CR-POMCP* ($p < 0.01$) con o sin peatones. Entre el *Indepependent HB-CR-POMCP* y el *Adaptive HB-CR-POMCP Follower*, se ha encontrado que el primero es el mejor para los dos mapas ($p < 0.01$).

4.3 EXPERIMENTOS REALES

El método presentado ha sido evaluado durante unos días de experimentación cerca de la facultad FME (figura 1) con los robots Tibi y Dabo [5] y unos voluntarios. El robot seguía la persona a una distancia de 1 m y entre ellos se mantenían una distancia de 1 m.

La tabla 2 muestra los valores de los parámetros, como ocurrieron muchos falsos positivos se ha incrementado la probabilidad de falsos negativos hasta $p_{\text{fn}} = 0.4$ y se han incrementado el número

Figura 3: Tres diferentes escenas donde se han hecho la búsqueda y el seguimiento de una persona. Los mapas de la segunda fila muestran los robots (azul y naranja) y las trayectorias ejecutadas; el círculo rojo indica la detección de la persona. La última fila muestra los mapas con el *belief* de Tibi y Dabo: en negro los obstáculos, los círculos azules son los robots y de blanco a rojo indica la probabilidad de que la persona esté ahí, y azul claro indica una probabilidad de 0.

de estados del *belief* a $n_{\text{belief}} = 5000$, y el número de simulaciones a $n_{\text{sim}} = 25000$.

Se grabaron alrededor de 10 experimentos (figura 3), cada robot navegó alrededor de unos 100 m y las personas (medido por los robots) caminaron unos 250 m. La distancia media era 3.1 ± 1.5 m (promedio $\pm$ desviación estándar), que es un poco más que el promedio 2.7 m en [8], pero esto es porque aquí tenemos dos robots siempre manteniendo una distancia mínima entre ellos. La persona fue visible al robot durante 66% del tiempo, mientras que en el método de un robot [8] fue visible sólo durante el 48% del tiempo. Los robots no usaron la información del otro robot durante 19% del tiempo por falla de conexión o por no haber usado la detección dentro de un tiempo máximo (3 segundos).

La Figura 3 muestra diferentes grabaciones de los experimentos; en la parte superior se muestran tres fotos y en la parte inferior hay una visión conjunta del experimento donde se muestra un mapa con los dos robots, sus trayectorias, y el mapa del *belief* de cada robot. El mapa del *belief* muestra que cuando una persona fue detectada, la posición era relativamente precisa, pero después de no haberla detectado durante cierto tiempo la probabilidad se propaga (centro). Aunque se han hecho los experimentos en un entorno pequeño, ya se puede ver claramente la cooperación entre los robots. Por ejemplo, en el caso de la figura 1 los robots empiezan en la esquina, pero tenían que mirar detrás del obstáculo. Se puede encontrar más información y vídeos en: `http://www.iri.upc.edu/people/agoldhoorn/ja2016/`.

5 CONCLUSIONES

En este trabajo se ha presentado un método nuevo para la localización cooperativa de una persona utilizando dos robots móviles en un entorno urbano. El método descrito, *Cooperative HB-CR-POMCP*, se basa en el trabajo anterior: *Adaptive HB-CR-POMCP Follower* el cual funciona en entornos grandes, en tiempo real, con estados contínuos y con obstáculos dinámicos. Se ha mostrado claramente que con el método presentado se puede buscar y seguir a una persona. La búsqueda también es más eficiente porque exploran el área donde la persona puede estar oculta de una manera estratégica teniendo en cuenta la probabilidad que esté la persona, la distancia hacia la meta y si el otro robot se dirige a esa posición. Además, el método es robusto porque funcione también si no hay comunicación.

El nuevo enfoque ha sido probado intensivamente en simulación, ha sido demostrado que funciona mejor en términos de distancia promedio a la persona, y que hay una visibilidad mejor que el método anterior: *Adaptive HB-CR-POMCP Follower*. Se han hecho varios experimentos reales con los robots Tibi y Dabo, los cuales muestran un comportamiento cooperativo en la búsqueda y el seguimiento. Se planean hacer más experimentos en entornos urbanos más grandes con otras personas paseando por el entorno. En el futuro se generalizará el método a un equipo de n robots buscando a una persona cooperativamente.

Agradecimientos

Este trabajo ha sido parcialmente financiado por el Ministerio de Ciencia e Innovación a través del proyecto RobTaskCoop(DPI2010-17112).

Referencias

[1] Amor-Martinez, A., A. Ruiz, F. Moreno-Noguer, and A. Sanfeliu (2014). On-board real-time pose estimation for uavs using deformable visual contour registration. In *Proc. of The International Conference in Robotics and Automation (ICRA)*.

[2] Burgard, W., M. Moors, C. Stachniss, and F. E. Schneider (2005). Coordinated multi-robot exploration. *IEEE Transactions on Robotics 21*(3), 376–386.

[3] Casper, J. and R. Murphy (2003). Human-robot interactions during the robot-assisted urban search and rescue response at the world trade center. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 33*(3), 367–385.

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

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

[6] Garrell, A., M. Villamizar, F. Moreno-Noguer, and A. Sanfeliu (2013). Proactive behavior of an autonomous mobile robot for human-assisted learning. In *IEEE Ro-man - International Symposium on Robot and Human Interactive Communication*, pp. 107–113.

[7] Gerkey, B. P. and M. J. Mataric (2002). Sold!: Auction methods for multirobot coordination. *IEEE Transactions on Robotics and Automation 18*(5), 758–768.

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

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

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

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

[12] Kawamura, K., R. Pack, M. Bishay, and M. Iskarous (1996). Design philosophy for service robots. *Robotics and Autonomous Systems 18*(1-2), 109–116.

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

[14] Lidoris, G., F. Rohrmuller, D. Wollherr, and M. Buss (2009). 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*, pp. 1416–1422.

[15] Ong, S. C. W., S. W. Png, D. Hsu, and W. S. Lee (2010, May). Planning under Uncertainty for Robotic Tasks with Mixed Observability. *The International Journal of Robotics Research 29*(8), 1053–1068.

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

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

[18] Trevai, C., Y. Fukazawa, J. Ota, H. Yuasa, T. Arai, and H. Asama (2003). Cooperative exploration of mobile robots using reaction-diffusion equation on a graph. In *Proceedings of the IEEE International Conference on Robotics and Automation*, Volume 2, pp. 2269–2274.

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

[20] Yamauchi, B. (1998). Frontier-based exploration using multiple robots. In *Proceedings of the second international conference on Autonomous agents*, pp. 47–53. ACM.

[21] Zinn, M., O. Khatib, B. Roth, and J. K. Salisbury (2004). Playing it safe [human-friendly robots]. *IEEE Robotics & Automation Magazine 11*(2), 12–21.