Robust Person Guidance by using Online POMDPs

Luis Merino[1], Joaquín Ballesteros[2], Noé Pérez-Higueras[1], Rafael Ramón Vigo[1], Javier Pérez-Lara[1], and Fernando Caballero[2]

[1] Robotics, Vision and Control Group
Universidad Pablo de Olavide, Sevilla, Spain,
[2] Robotics, Vision and Control Group
Universidad de Sevilla, Spain

Abstract. The paper considers a guiding task in which a robot has to guide a person towards a destination. A robust operation requires to consider uncertain models on the person motion and intentions, as well as noise and occlusions in the sensors employed for the task. Partially Observable Markov Decision Processes (POMDPs) are used to model the task. The paper describes an enhancement on online POMDP solvers that allow to apply them to larger problems. The algorithm is used to control the robot in real-time for the guiding application. Results in simulation illustrate the approach.

Keywords: robust navigation, POMDP

1 Introduction

In the last years, there has been an impressive development on localization, mapping, SLAM and navigation techniques. The scale of the scenarios to which this techniques can be applied has increased dramatically. But despite all these advances, achieving long-term autonomy, either in space and time, requires to enhance the robustness of all these skills. For this, a robot should cope with imperfect information, failure modes, partial models, etc.

In particular, we are interested in the robust operation of robots in human inhabited spaces. The FROG FP7 project[3] aims to deploy a guiding robot in touristic sites involving outdoor and partially outdoor scenarios. While robot guides has been developed since more than a decade [26, 19], the project considers as new contributions the development of social behaviors and their adaptation by integrating social feedback, as well as the robust operation in outdoors crowded scenarios. Furthermore, it aims to demonstrate the operation of the robot for two weeks at the Royal Alcazar in Seville (see Fig. 1).

Among other activities, one particular application considered in the project is person guidance. Robustly navigating in these crowded scenarios (the Royal Alcazar may have around 5000 visits per day) requires to consider issues due

[3] http://www.frogrobot.eu

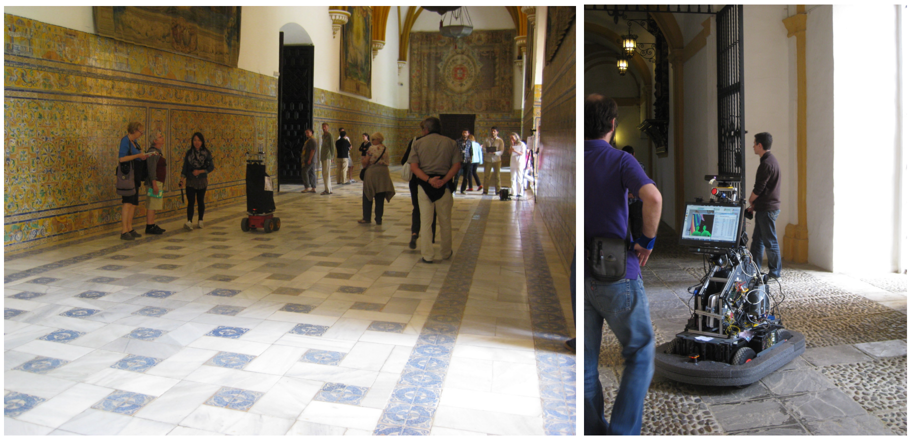

Fig. 1: The FROG project aims to deploy a guiding robot with a fun personality, considering social feedback, in the Royal Alcazar of Seville and the Zoo of Lisbon. Two pictures of the first scenario are presented here. On the right, and initial design of the robot components can be seen.

to the high dynamics of the environment. Furthermore, guiding a person in this scenario involves not only ensuring a safe and efficient navigation but also social interaction and social awareness when achieving these goals.

For this and other applications, a robust operation, like robust navigation, requires reasoning about all the potential uncertainties present on the robotic system, due to imperfect models, limited information, errors, noise, etc. One framework that allows considering uncertainties in a principled way is Partially Observable Markov Decision Processes (POMDPs) [8]. By using POMDPs, the uncertainties in the sensors, as well as uncertainties on the models employed by the robot, are considered when planing the actions that the robot has to execute.

However, the broader application of planning under uncertainties to robotic systems requires to develop methods that cope with the curse of dimensionality: in this case the planning problems are posed not on the state space, but on the much harder information space (the space of all potential actions and observations histories that the robot may perform/gather).

In the paper, we present firstly a method for alleviating the complexity of online POMDPs, allowing to apply them to larger problems. Then, it is described how this method can be used to model the task of person guidance. Results in simulation show the applicability of the approach. The paper finalizes with some conclusions.

1.1 Related work

POMDPs are increasingly used in robotics [11, 14, 6, 4]. This is due to the development of more efficient offline POMDP solvers in the last decade [23, 9, 3].

However, their broader application to robotics has been precluded due to curse of dimensionality. Besides offline solvers, online POMDPs have been also considered [18, 6], as they are more suitable for certain robotic applications. Still they are affected by the same issues.

Efficient and robust navigation in crowded environments requires taking into account human behavior models, social constraints and their uncertainties[7]. On one hand, besides the static obstacles the motion of the persons have to be considered. Typical approaches to human motion prediction simply assume a constant velocity, which is not valid in most cases. In general, human navigation intent will depend on the function and structure of the environment [25, 2, 12].

Furthermore, the interaction of the robot and the humans has to be taken into account to allow for an efficient navigation[27]. In the particular application of person guidance, in [5] probabilistic models of human-interaction are extracted from data for the purposes of person guiding. Finally, applications in which persons are also involved require to consider social constraints, like the human commitment and goals. Related to the work presented here, the authors in [16] employ Markov Decision Processes to predict the destination of the persons in guiding applications. However, the uncertainties on the observation processes are not considered.

Closest to our work, in [24] a POMDP is also used to infer the intentions of the person for wheelchair navigation. Similar ideas are considered, but in a different scenario. Moreover, offline POMDP models are used, so the system has to be re-planned offline if a different scenario is considered. Our online system allow to build the models on the fly. Furthermore, the proposed system is able to work with a larger scenario in terms of state space and observation outcomes.

2 POMDPs in a Nutshell

As commented above, in this paper we will analyze the use of POMDPs to model navigation tasks like person guiding. Formally, a discrete POMDP is defined by the tuple $\langle S, A, Z, T, O, R, D, \gamma \rangle$ [8]. The *state space* is the finite set of possible states $s \in S$; the *action space* is defined as the finite set of possible actions $a \in A$ that the robot can perform; and the *observation space* consists of the finite set of possible observations $z \in Z$. At every step, an action is performed by the robot, an observation is made and a reward is given.

After performing an action a, the state transition is modeled by the conditional probability function $T(s', a, s) = p(s'|a, s)$, allowing to model failures and uncertainties on the prediction models. In the same way, measurement process is modeled by the conditional probability function $O(z, a, s') = p(z|a, s')$, which permits to consider non-observability, occlusions, noise, etc.

The reward obtained at each step is $R(s, a)$. The planning objective is to maximize the sum of expected rewards, or *value*, for a planning horizon of D time steps. To ensure that the sum is finite when $D \to \infty$, rewards are weighted by a discount factor $\gamma \in [0, 1)$.

POMDPs consider that the state is non-observable; therefore, a belief function b is maintained by using Bayes rule. The belief obtained if we apply the action a and get the observation z is $b'(s') = \tau(b, a, z) = \eta O(z, a, s') \sum_{s \in S} T(s', a, s) b(s)$. The normalization constant:

$$\eta = P(z|b, a) = \sum_{s' \in S} O(z, a, s') \sum_{s \in S} T(s', a, s) b(s) \tag{1}$$

gives the probability of obtaining a certain observation z after executing action a for a belief b. As said above, the objective is to plan a policy which indicates the action that has to be performed given the information available (summarized by the belief) $a = \pi(b)$ and that maximizes the cumulative expected reward, or value $V^\pi(b)$:

$$V^\pi(b) = R(b, \pi(b)) + \gamma \sum_{z \in Z} P(z|b, a) V^\pi(b^z_{\pi(b)})) \tag{2}$$

where $R(b, a) = \sum_s R(s, a) b(s)$ is the expected immediate reward[4]. The value of the optimal policy is usually denoted by $V^*(b)$ and associated to it is the optimal Q function:

$$Q^*(b, a) = R(b, a) + \gamma \sum_{z \in Z} P(z|b, a) V^*(b^z_a)) \tag{3}$$

2.1 POMDP Solver

Modeling a particular robotic task, like person guidance, implies to define the transition function $T(s', a, s)$ and the observation model $O(z, a, s')$. Notice that both models consider uncertainties, either in the outcome of actions and in the observation process. These models can be defined by hand or they can be learnt from data [5].

Once the models are defined, the task is codified into the POMDP by designing an appropriate reward function $R(s, a)$, which indicate the expected behavior of the robot. Again, this reward function can be defined by hand, or could be learnt from data [7].

Given the reward and the models, the optimal policy can be obtained by computing the policy $\pi(b)$ that maximizes (2) using dynamic programming. However, POMDPs are quite hard to solve (they are PSPACE-complete [15]). Current offline solvers, like [23, 9, 3], apply some approximations to obtain a solution. Furthermore, they try to solve the dynamic programming problem over the full belief state space. These algorithms require to recompute the full policy when there are changes on the environment dynamics or the sensor models.

As commented above, online solvers are more suitable for some robotic tasks. An online POMDP solver tries to determine the optimal action for the current belief point b, and not for all the possible beliefs. To achieve this, it creates

[4] $b^z_a = \tau(b, a, z)$ will be used to obtain a more compact formula.

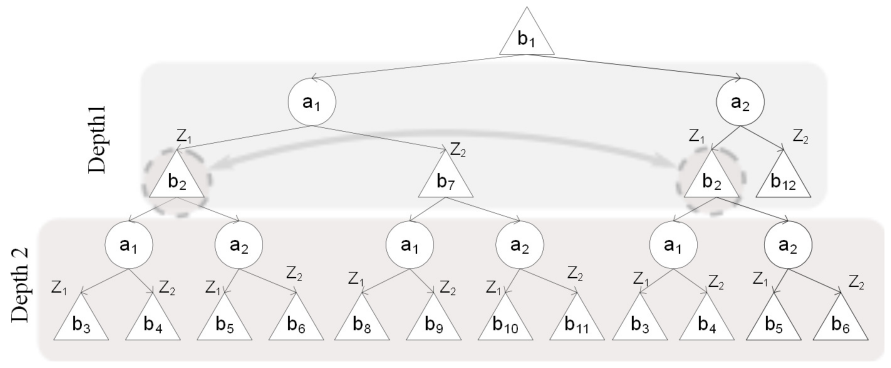

Fig. 2: An AND-OR Belief Tree with 2 actions and 2 observations. The OR-node (where the robot has to take an action) are represented by triangles and the AND-nodes by circle (when observations, which are not controlled, are received). If a repeated belief appears at the same depth, it means that it has an identical subtree and the same value (like belief b2).

an AND-OR tree by exploring the next beliefs for all possible actions and for all possible observations starting at the initial belief (see Fig. 2). The dynamic programming solution to (2) is applied from the leaves to the root, obtaining the best action.

This tree is created on demand, and can be adapted online if the models or the reward function change. As we will see, this allows us to model tasks as the guiding task described below. Furthermore, this kind of structures can be used to create anytime algorithms [17], in which the time available is used to explore the tree as deep as possible.

The branching factor in AND-OR POMDP trees is $|A||Z|$ where $|A|$ is the number of actions and $|Z|$ is the number of observations, and the number of leaves nodes for a tree of depth D is $(|A||Z|)^D$, that is, exponential on the planning horizon. Exploring all the nodes to obtain the optimal action is, therefore, not a option. In [18], a classification of POMDP online algorithms can be found, and also three strategies that are employed to improve the computing time required to choose the best action:

– Monte Carlo sampling algorithms: minimize the branching factor by sampling a subset of observations.
– Heuristic search algorithms: guide the search of the most relevant branch nodes.
– Branch and Bound algorithms: Prune nodes that are suboptimal compared to other that have already been expanded.

We have developed an additional improvement that can be applied with any of the previous strategies. The main idea is to take advantage of the true topology of the belief space, which is actually a graph instead of a tree. For this, we introduce a pseudo-metric into the belief space to determine if two belief points are actually the same, that is, if we are revisiting a belief point, which then does not have to be expanded again within the AND-OR tree.

In this paper we employ a first idea in that direction. In the expansion of the AND-OR tree, if the next belief to be expanded is similar to a saved belief at the same depth, that belief will not be expanded because its value is already calculated (see Fig. 2). The similarity between two belief points in the tree b and b' is computed by using the Jensen-Shannon divergence $D_{JS}(b\|b')$ [1] (its square root is actually a true metric):

$$D_{JS}(b\|b') = \frac{1}{2}(D_{KL}(b\|\frac{b+b'}{2}) + D_{KL}(b'\|\frac{b+b'}{2})) \tag{4}$$

where

$$D_{\mathrm{KL}}(b\|b') = \sum_{s\in S} b(s) \ln \frac{b(s)}{b'(s)} \tag{5}$$

The Algorithm 1.1, called FSBS, proceeds via look-ahead search up to a fixed depth d. We use the structure of the RTBSS algorithm proposed by [18] to elaborate the FSBS, which therefore also prunes branches on the tree which are suboptimal by using lower and upper bounds on the optimal value function. In particular, the algorithm uses the max-planes lower bound [21] of the optimal value implemented in [22] (line 3). The δ function determines how the FSBS algorithm propagates this lower bound from the leaves up to the root.

$$\delta(b,0) = LowerBound(b) \tag{6}$$

$$\delta(b,d) = \max_{a\in A}(R(b,a) + \gamma \sum_{z\in Z} Pr(z|b,a)\delta(b_a^z, d-1)) \tag{7}$$

To determine the similarity between beliefs, we keep a node list for each depth d, $nodeList_d$. Every node contains the following items: the belief b; an action; and the accumulated reward if that action is applied to the belief, $\delta(b,d)$. We only keep the beliefs up to depth $D-1$ because the leaf nodes cannot be expanded.

The function *orderbyAccReward* in line 5 is used to find the accumulated rewards that are obtained when we apply each action to the current belief.

For each action at depth d, the *orderbyAccReward* function looks for a similar belief in $nodeList_d$, b', considering a similarity threshold *th*. If a belief is successfully found, the accumulated reward that was already stored, $\delta(b',d)$, is assigned to it, and if not, it is assigned as ∞ in order to get expanded first. The *orderbyAccReward* function returns a list, sorted by accumulated reward, in which each element contains the following items: the action associated; the accumulated reward if this action is applied to the input belief if exists; and a variable that indicates whether the resulting belief is already in a depth on the tree or not.

In line 12-13 we reuse the accumulated reward if a similar node is found at the same depth. In this case, we stop expanding along that path. If not, we expand

Algorithm 1.1: FSBS Algorithm

```
 1: function FSBS($b, d, th$)
 2:     if $d == 0$ then
 3:         return $LowerBound(b)$
 4:     end if
 5:     $\{st_1, st_2, ..., st_{|A|}\} \leftarrow orderbyAccReward(b, d, th)$
 6:     $L_T(b) \leftarrow -\infty$
 7:     $i \leftarrow 0$
 8:     while $i < |A|$ $AND$ $st_i.AccReward > L_T(b)$ do
 9:         $a \leftarrow st_i.IdAction$
10:         $rAcc \leftarrow st_i.AccReward$
11:         $L_T(b, a) \leftarrow -\infty$
12:         if $st_i.isFoundSimilar$ then
13:             $L_T(b, a) \leftarrow Reward(b, a) + \gamma rAcc$
14:         else
15:             $L_Z(b, a) \leftarrow \sum_{z \in Z} P(z|b, a) FSBS(b_a^z, d-1, th)$
16:             $L_T(b, a) \leftarrow Reward(b, a) + \gamma L_Z(b, a)$
17:             $saveNode(b, a, d, L_Z(b, a))$
18:         end if
19:         $L_T(b) \leftarrow \max\{L_T(b), L_T(b, a)\}$
20:     end while
21:     return $L_T(b)$
22: end function
```

and keep the node (line 15-18). To finish we choose the action that maximizes the accumulated reward (line 21).

We calculate the optimal policy as:

$$\pi^*(b, D) = \arg\max_{a \in A}(R(b, a) + \gamma \sum_{z \in Z} P(z|b, a) FSBS(b_a^z, D-1, th)) \tag{8}$$

This is applied in each planning iteration. At every iteration, the optimal action is applied and then a new forward search is performed, in a receding horizon fashion.

The algorithm has been tested in benchmark problems showing that using this similarity measurements it is possible to obtain a similar expected solution by expanding several orders of magnitude less nodes [1] than state of the art algorithms. In particular, Fig. 3 shows the comparison of the original RTBSS algorithm and the FSBS algorithm for the RockSample benchmark [20]. More details can be found at [1].

3 Modeling the guiding task as a POMDP

In a guiding application, the robot has to guide a person or group of persons towards a common destination. The typical solution to the problem is to plan a

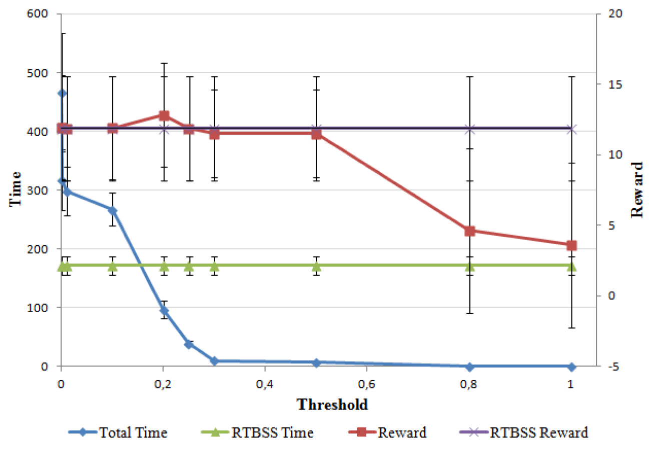

Fig. 3: Expected accumulated reward (right vertical axis) and execution time (left vertical axis) for RTBSS and FSBS using the JS divergence (RockSample). Different thresholds in the JS divergence (horizontal axis) are considered to determine if two belief points are the same in the tree. The larger the thresholds the less nodes are expanded. It can be seen that for thresholds between 0.3 and 0.5 in the similarity the expected reward is very similar, but the execution time is two orders of magnitude lower.

path towards the destination and then follow the path by controlling the speed of the robot by using some feedback on the person being guided, like laser, visual information or a combination of them [13].

This requires having an estimation of the position of the person. However, imperfect sensors, false positives, occlusions, etc, may lead to uncertainties on the person position. In particular, vision algorithms can be affected by illumination changes, which makes it hard to track robustly a person being guided.

Moreover, one of the main sources of uncertainty in this problem is that the person may change his/her mind, or decide to stop for a while at a different interest point in his way to the destination. From a social point of view it is important that the robot considers the person goals while guiding the person, in order to wait for the person or even to change its goal accordingly. However, the robot cannot have a direct observation on this person goal.

As assumptions, we will assume that the robot has a map of the scenario. This map also includes a model of potential interest points for persons, determined beforehand by using information about the typical tours performed by tourists. This map can be also learnt by using data gathered by the robot [12].

Once the person selects a destination, a path planner determines a path towards this destination. In order to model the problem by using the POMDP framework described above, this path is then discretized into a set of points with a granularity that can be adjusted (1.5 meters in the current implementation). The map of the scenario is analyzed to discover if in the way towards the final

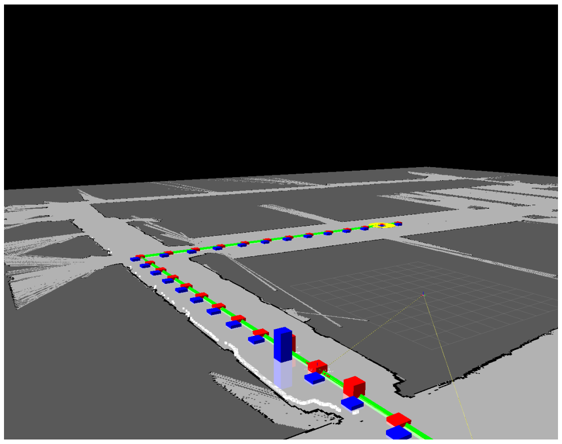

Fig. 4: The model employed: the planned path (in green) is discretized. At each point, the bars blue and red indicate the estimated belief on the robot and person positions respectively. Furthermore, the yellow circles indicate the current robot belief on the person goal. The sizes are proportional to the marginal probabilities.

destination there are points that may be of interest for the person and are considered as intermediate goals.

The state space S is then composed by the position of the robot and the person within the discretized path, as well as the estimated goal of the person. The robot will maintain, thus, a belief state (a probability distribution) over its pose, the person position and the person intention (see Fig. 4).

The belief state is initialized to the initial position of robot and person in the tour, and the goal is initially set as the destination selected (the end of the path).

3.1 Observation model

The robot sensors considered are the output of the localization system and a camera for person tracking.

The localization of the robot is provided by a map-based Monte-Carlo localization algorithm using the laser data of the robot. By using this algorithm, the robot can know its position within the path. The accuracy of the algorithm for different parts of the map is known beforehand and is included into the POMDP model. This model assigns some probability of obtaining a robot position measurement in the surroundings of the real position (for instance, in the corridor that can be seen in Fig. 4, the robot has some uncertainty in its position on the direction of the corridor because of the symmetry of the scenario; this can be included into the observation model).

For person guidance, a visual tracker is employed. The tracker is able to cope with illumination changes and it is quite robust [10]. However, the current robot

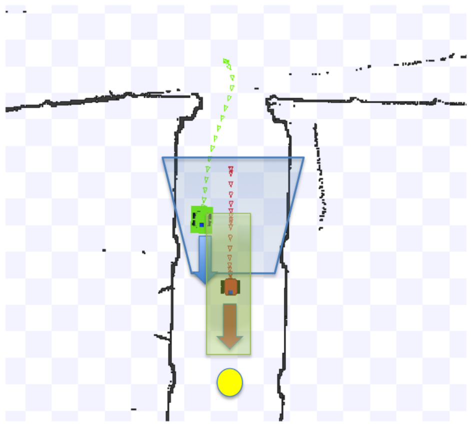

Fig. 5: Prediction models (arrows): the robot (red) and the person (green) can move forward or stay. If the robot decides to proceed forward it will reach the next point on the trajectory with certainty. The motion of the person is uncertain and will also depend on the presence of intermediate goals (yellow). Observation models: the robot can detect with its camera if the person is within its field of view (blue polygon) with high probability. This field of view is about 3 meters long. The position of the robot is obtained by the localization system with a precision depending on the place in the map (green rectangle).

system cannot orient the camera independently. Furthermore, it is only able to track the person if it is below a certain distance threshold from the camera. On the other hand, it is able to recognize persons that has already seen before, so if a person goes out and back into the field of view it will recognize him.

Therefore, the model considers a high probability of detection when the person is in the field of view of the camera (see Fig. 5), while accounting for a small possibility of misdetections.

3.2 Prediction function

The action space of the robot considered here is to continue to the next discrete point on the trajectory or to wait for the person (it controls the speed in a on/off fashion, although this is implemented smoothed in the velocity controller).

The robot actions are modeled as deterministic (that is, the robot will reach the next point if commanded so, even though the possibility of crowded places in the scenario may be modeled).

The motion of the person is uncertain, though. The model employed considers that the person will adapt to the robot pace, but he may wander a bit, so there is some small probability that the person stays at his current position. Furthermore, the intentions of the person will depend on the place he is transversing. If the person is close to an intermediate goal, then the person may decide to stop by.

Table 1: Comparative table for a planning depth of 5

Algorithm	Number of nodes	Time (s.)	Averaged Reward
RTBSS	7776	6.75	241.52
FSBS (threshold 0.3)	108	0.1	238.34

Therefore, in those cases there is a larger probability of staying and changing the person goal.

3.3 Design of the reward function

The main design choice in a POMDP is the reward function, with which we encode the desired behavior of the robot. For this task, the objective is to follow the path, guiding the person and adapting to the person motion and intentions. Furthermore, it is important not to lose the person.

Therefore, the robot receives a positive reward if it goes towards the goal of the person and maintains the person below a certain distance. A larger reward is received if the final destination is reached than for the intermediate goals. However, the robot receives a penalization if it proceeds towards a different goal than that of the person (the person may stop for a while in one of the intermediate goals). This requires to reason about the potential intentions of the persons.

Besides, the actions are penalized so faster achievements are preferred.

4 Simulations

In order to analyze the modeling of the task by using POMDPs, in this section we present some results obtained by applying the mentioned algorithm in simulations. In the simulation, the robot is controlled by the algorithm, while the person is simulated by a robot teleoperated by a person. Figure 6 shows the typical execution of the algorithm.

In the example, the trajectory selected is discretized into 27 zones, in which 2 intermediate goals, besides the final destination, are identified, giving a state space of dimension $|S| = 2187$ $(27 \times 27 \times 3)$, with $|A| = 2$ potential actions and $|O| = 54$ potential observations (the position of the robot and if the person is see or not by the camera tracker). An AND-OR tree for this problem and depth 5 contains more than 10^{10} nodes.

Table 1 shows a comparison of the number of nodes and reward obtained by different methods and a planning depth of 5. As it is seen, the presented algorithm can run in real-time, obtaining a similar reward as the one obtained by the algorithm RTBSS [18].

Fig. 7 describe some of the situations encountered in the execution. It can be seen how the system is able to reason about its limitations in its sensor data and

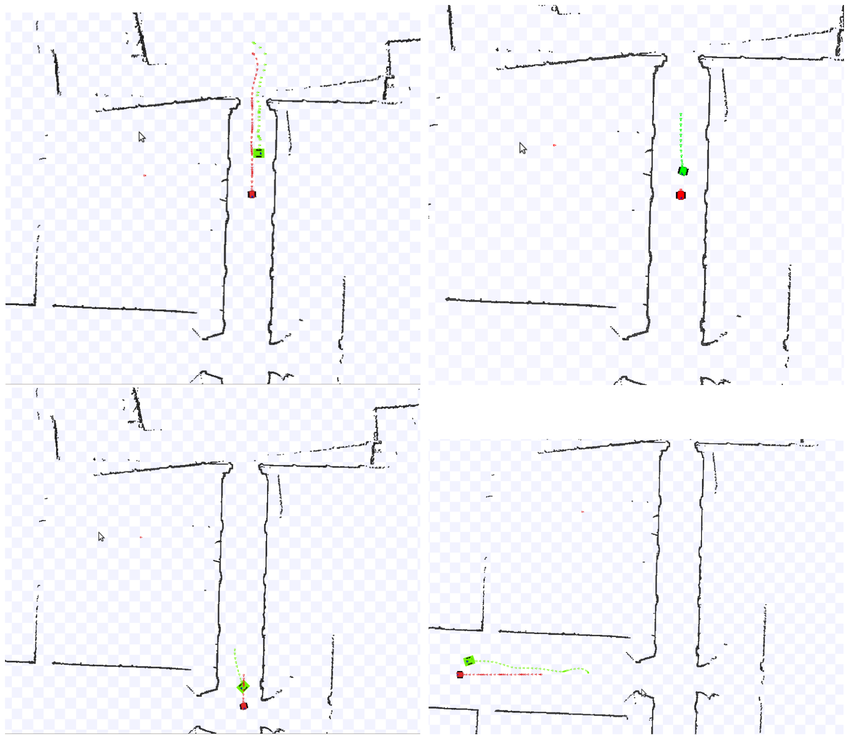

Fig. 6: Typical execution of the POMDP controller guiding the person (green) towards the destination.

in the models in order to adapt itself to the situation. In the set of simulations, the average distance between the robot and the person is of 2.04 meters.

5 Conclusions

The paper has presented the application of POMDPs to navigation tasks, in particular a robot guiding task. POMDPs are a way of reasoning about the uncertainties of the system when controlling the robot, which allows to enhance the robustness of the robot operation. In order to apply these methods to robotics, it is needed to develop techniques able to cope with larger state, action and observation spaces. The paper presents one method to alleviate the complexity of POMDPs. Using this method it is possible to apply a POMDP model to the guiding task in real-time.

As a future work, we will perform actual experiments in a guiding setup, and compare the technique with other approaches for guiding [16].

Furthermore, the FSBS method will be extended to build belief graphs, which should reduce further the number of expanded nodes, allowing to cope with larger problems. Furthermore, nothing precludes to use the same ideas for continuous representations of the belief space, as well as considering continuous actions, which is very relevant for robotic applications.

Regarding the use of POMDPs for navigation tasks in pedestrian environments, one of the main issues is the design of the reward functions to define the particular task. A more interesting approach is to learn the reward function from demonstration or examples from humans [7]. This will also allow to consider social behaviors into the navigation stack, by transferring the way human guides behave to the robot.

6 Acknowledgments

This work is partially funded by the European Commission 7th Framework Programme under grant agreement no. 288235 (FROG), and the regional project ROMOCOG (P10-TEP-6412, P09-TEP-4479).

References

1. Ballesteros, J., Merino, L., Trujillo, M.A., Viguria, A., Ollero, A.: Improving the efficiency of online POMDPs by using belief similarity measures. In: Proc. International Conference on Robotics and Automation, ICRA (2013)
2. Bennewitz, M., Burgard, W., Cielniak, G., Thrun, S.: Learning motion patterns of people for compliant robot motion. I. J. Robotic Res. 24(1), 31–48 (2005)
3. Bonet, B., Geffner, H.: Solving pomdps: Rtdp-bel vs. point-based algorithms. In: Proceedings of the 21st international jont conference on Artifical intelligence. pp. 1641–1646. IJCAI'09, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2009)
4. Capitan, J., Spaan, M., Merino, L., Ollero, A.: Decentralized Multi-Robot Cooperation with Auctioned POMDPs. The International Journal of Robotics Research 32, 650–671 (2013)
5. Feil-Seifer, D., Mataric, M.: People-aware navigation for goal-oriented behavior involving a human partner. In: Proceedings of the IEEE International Conference on Development and Learning (ICDL) (2011)
6. He, R., Bachrach, A., Roy, N.: Efficient planning under uncertainty for a target-tracking micro-aerial vehicle. In: Proc. International Conference on Robotics and Automation, ICRA (2010)
7. Henry, P., Vollmer, C., Ferris, B., Fox, D.: Learning to navigate through crowded environments. In: ICRA'10. pp. 981–986 (2010)
8. Kaelbling, L.P., Littman, M.L., Cassandra, A.R.: Planning and acting in partially observable stochastic domains. Artificial Intelligence 101, 99–134 (1998)
9. Kurniawati, H., Hsu, D., Lee, W.: SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. In: Proceedings of the Robotics: Science and Systems Conference. Zurich, Switzerland (2008)
10. Liwicki, S., Tzimiropoulos, G., Zafeiriou, S., Pantic, M.: Efficient online subspace learning with an indefinite kernel for visual tracking and recognition. IEEE Transactions on Neural Networks and Learning Systems 23, 1624–1636 (October 2012)
11. López, M., Bergasa, L., Barea, R., Escudero, M.: A navigation system for assistant robots using visually augmented POMDPs. Autonomous Robots 19(1), 67–87 (2005)

12. Luber, M., Tipaldi, G.D., Arras, K.O.: Place-Dependent People Tracking. International Journal of Robotics Research 30(3) (March 2011)
13. Merino, L., Gilbert, A., Capitan, J., Bowden, R., Illingworth, J., Ollero, A.: Data Fusion in Ubiquitous Networked Robot Systems for Urban Services. Annals of Telecommunications, Special Issue Ubiquitous Robots 67 (2012)
14. Ong, S., Png, S.W., Hsu, D., Lee, W.S.: POMDPs for Robotic Tasks with Mixed Observability. In: Proc. Robotics: Science and Systems, RSS (2009)
15. Papadimitriou, C., Tsitsiklis, J.N.: The complexity of markov decision processes. Mathematics of Operations Research 12(3), 441–450 (1987)
16. Perrin, X., Colas, F., Pradalier, C., Siegwart, R.: Learning to identify users and predict their destination in a robotic guidance application. In: Howard, A., Iagnemma, K., Kelly, A. (eds.) Field and Service Robotics, Springer Tracts in Advanced Robotics, vol. 62, pp. 377–387. Springer Berlin Heidelberg (2010)
17. Pineau, J., Gordon, G., Thrun, S.: Point-based value iteration: An anytime algorithm for POMDPs. In: Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI). IJCAI, Acapulco, Mexico (2003)
18. Ross, S., Pineau, J., Paquet, S., Chaib-draa, B.: Online planning algorithms for POMDPs. Journal of Artificial Intelligence Research (2008)
19. Siegwart, R., Arras, K.O., Bouabdallah, S., Burnier, D., Froidevaux, G., Greppin, X., Jensen, B., Lorotte, A., Mayor, L., Meisser, M., Philippsen, R., Piguet, R., Ramel, G., Terrien, G., Tomatis, N.: Robox at Expo.02: A large-scale installation of personal robots. Robotics and Autonomous Systems 42(3-4), 203–222 (March 2003)
20. Smith, T., Simmons, R.: Heuristic search value iteration for POMDPs. In: Proceedings of the 20th conference on Uncertainty in artificial intelligence. pp. 520–527. AUAI Press (2004)
21. Smith, T.: Probabilistic Planning for Robotic Exploration. Ph.D. thesis, The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA (July 2007)
22. Smith, T.: ZMDP software for POMDP and MDP planning. http://www.cs.cmu.edu/ trey/zmdp/ (2012)
23. Spaan, M.T.J., Vlassis, N.: Perseus: Randomized point-based value iteration for POMDPs. Journal of Artificial Intelligence Research 24, 195–220 (2005)
24. Taha, T., Miro, J., Dissanayake, G.: Pomdp-based long-term user intention prediction for wheelchair navigation. In: Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on. pp. 3920–3925 (2008)
25. Thompson, S., Horiuchi, T., Kagami, S.: A probabilistic model of human motion and navigation intent for mobile robot path planning. In: Gupta, G.S., Mukhopadhyay, S.C. (eds.) ICARA. pp. 663–668. IEEE (2009)
26. Thrun, S., Beetz, M., Bennewitz, M., Burgard, W., Cremers, A.B., Dellaert, F., Fox, D., Hahnel, C.: Probabilistic algorithms and the interactive museum tour-guide robot minerva. The International Journal of Robotics Research 19, 972–999 (October 2000)
27. Trautman, P., Krause, A.: Unfreezing the robot: Navigation in dense, interacting crowds. In: IROS. pp. 797–803. IEEE (2010)

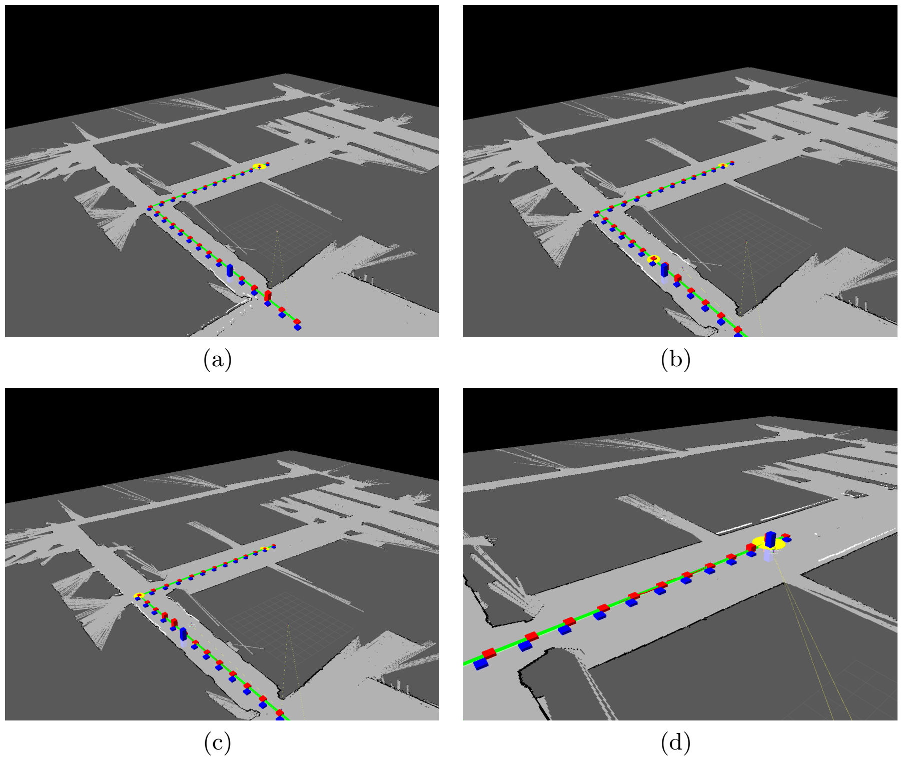

Fig. 7: a) The initial situation: the estimated goal (in yellow) is the initial destination. b) The robot and the person (the belief on their positions in blue and red, respectively) are approaching a potential intermediate goal. Therefore, the robot puts some probability mass on this goal (yellow), as well as the one at the final destination. This provokes that the robot slows down, as it considers that the person may stay at the intermediate goal. c) The person stays for a while in the intermediate goal, but then proceeds. When doing this he goes out of the field of view of the camera. However, the models and the Bayes filter allow to use this negative information to infer the motion of the person and the robot proceeds towards the final destination again. The belief on the potential goals change (yellow). d) The robot and person reach the final destination.