Real-Time Hierarchical POMDPs for Autonomous Robot Navigation

Amalia Foka and Panos Trahanias

Technical Report

ICS-TR347

Institute of Computer Science

Foundation for Research and Technology – Hellas (FORTH)

P.O.Box 1385, Heraklion, 711 10 Crete, Greece

January 2005

Abstract

This paper proposes a new hierarchical formulation of POMDPs for autonomous robot navigation that can be solved in real-time, and is memory efficient. It will be referred to in this paper as the Robot Navigation - Hierarchical POMDP (RN-HPOMDP). The RN-HPOMDP is utilized as a unified framework for autonomous robot navigation in dynamic environments. As such, it is used for localization, planning and local obstacle avoidance. Hence, the RN-HPOMDP decides at each time step the actions the robot should execute, without the intervention of any other external module for obstacle avoidance or localization. Our approach employs state space and action space hierarchy, and can effectively model large environments at a fine resolution. Finally, the notion of the *reference* POMDP is introduced. The latter holds all the information regarding motion and sensor uncertainty, which makes the proposed hierarchical structure memory efficient and enables fast learning. The RN-HPOMDP has been experimentally validated in real dynamic environments.

1 Introduction

The autonomous robot navigation problem has been studied thoroughly by the robotics research community over the last years. Contemporary methods for robot navigation [11, 16, 8, 9] do not considerably take into account the robot motion uncertainty and sensor uncertainty. Yet, incorporating uncertainty in methods for planning is crucial to their performance. Uncertainty can cause to direct the robot in executing false actions. Probabilistic methods that integrate uncertainty in motion planning have not been well studied until now, in contrast to probabilistic methods for mapping and localization. In this paper we introduce a Hierarchical POMDP (HPOMDP) for probabilistic navigation. Our HPOMDP formulation simultaneously addresses probabilistically all aspects of navigation, that is motion planning, localization and local obstacle avoidance.

Partially Observable Markov Decision Processes (POMDP) provide the mathematical framework for probabilistic planning. POMDPs model the hidden state of the robot that is not completely observable and maintain a belief distribution of the robot's state. Planning with POMDPs is performed according to the belief distribution. Therefore, actions dictated by a POMDP drive the robot to its goal but also implicitly reduce the uncertainty of its belief.

Although POMDPs successfully meet their purpose of use, they are intractable to solve with exact methods when applied to real-world environments modelled at a fine resolution. Many approximation methods for solving POMDPs are present in the literature and have also been applied to robotics problems [1, 12, 21, 6, 22, 28, 4, 26]. Due to the involved computational complexity, these approximation methods can only deal with problems where the

size of the state space is limited to at most a few thousands states. As a result, all the approximation methods cannot model large real world environments at a fine resolution and hence POMDPs are mainly used as high level mission planners.

In this paper, we propose a hierarchical representation of POMDPs for autonomous robot navigation (RN-HPOMDP) that can effectively model large real world environments at a fine resolution. Moreover, the proposed RN-HPOMDP can be solved in real time. It is utilized as a unified framework for autonomous robot navigation, implying that no other external modules are used to drive the robot. RN-HPOMDP integrates the modules for localization, planning and local obstacle avoidance; it is solved on-line at each time step and decides the actual actions the robot performs.

In Section 2, the necessary theoretical background for POMDPs is given followed by the formulation of each element of a POMDP for the autonomous robot navigation problem in Section 3. In Section 4 the structure of the RN-HPOMDP is presented. The methodoly used for learning and planning with the RN-HPOMDP is presented in Sections 5 and 6, respectivelly.

Two other HPOMDP approaches are currently present in the literature that employ either state space hierarchy [27], applied as a high level mission planner, or action and state space hierarchy [18], applied for high level robot control and dialogue management. Independently and concurrently with these works, we have come up with a HPOMDP[1] that applies both state space and action space hierarchy. It is specifically designed for the autonomous robot navigation

[1]Preliminary versions of our HPOMDP are presented in [2, 3].

problem, hence the term RN-HPOMDP, and offers specific advantages over the two approaches mentioned above. A comparison between the RN-HPOMDP and the other previously mentioned approaches is presented in Section 7. Experimental results, presented in Section 8, have shown the applicability of the RN-HPOMDP for autonomous robot navigation in large real world and dynamic environments where humans and moving objects are effectively avoided and the robot follows optimal paths to reach its destination. Finally, this paper's conclusions and future work directions are presented in Section 9.

2 Partially Observable Markov Decision Processes (POMDPs)

POMDPs are a model of an agent interacting synchronously with its environment. The agent takes as input the state of the environment and generates as output actions, which themselves affect the state of the environment. In the POMDP framework, a system acting in the world is not guaranteed at any time to know the state of the world, i.e. which state of the environment it occupies. Hence, states are partially observable.

Formally, a POMDP is a tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \mathcal{Z}, \mathcal{O} \rangle$, where

- $\mathcal{S}$, is a finite set of all possible states of the environment that the agent might occupy and are partially observable.
- $\mathcal{A}$, is a finite set of actions.
- $\mathcal{Z}$, is a finite set of observations.

- $\mathcal{T} : \mathcal{S} \times \mathcal{A} \rightarrow \Pi(\mathcal{S})$ is the *state transition function*, giving for each state and agent action, a probability distribution over states. $\mathcal{T}(s, a, s')$ is the probability of ending in state s', given that the agent starts in state s and takes action a, $p(s'|s, a)$. The distribution over the state space depends only on the current state-action pair and not on previous state-action pairs. This requirement ensures the *Markov property* of the process.

- $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is the *reward function*, giving the expected immediate reward gained by the agent for taking an action a when it is in state s, $\mathcal{R}(s, a)$.

- $\mathcal{O} : \mathcal{A} \times \mathcal{S} \rightarrow \Pi(\mathcal{Z})$ is the *observation function* giving for each state and agent action, a probability distribution over observations. $\mathcal{O}(s', a, z)$ is the probability of observing z, in state s' after taking action a, $p(z|s', a)$.

2.1 Belief State

A POMDP agent is composed of two components [27]: the state estimator component and the policy component. The state estimator component maintains at all times the belief state b_t of the agent. The belief state is a discrete probability distribution over the set of environment states, $\mathcal{S}$, representing for each state the agent's belief that is currently occupying that state. Hence, $b_t(s)$ is the probability of the agent being in state s at time t, $p_t(s : s \in \mathcal{S})$. The set of all possible belief states is $\mathcal{B}$. The state estimator updates the belief state of the agent every time it executes an action based on the action it executed and the observation it perceived as explained in Section 2.2. The policy component maps a belief state to an optimal action as explained in Section 2.3.

2.2 Belief Update

The state estimator component of a POMDP updates the belief state of the agent every time it executes an action. Given the belief state of the agent at time t, b_t, we would like to compute the belief state at time $t+1$, b_{t+1}, after a transition in the process where the agent occupies state s, executes an action a and perceives an observation z. The belief that the agent is in the resulting state s' is derived by:

$$\begin{aligned} b_{t+1}(s') &= P(s'|z,a,b_t) \\ &= \frac{P(z|s',a,b_t)P(s'|a,b_t)}{P(z|a,b_t)} \\ &= \frac{P(z|s',a,b_t)\sum_{s\in S} P(s'|a,b_t,s)P(s|a,b_t)}{P(z|a,b_t)} \\ &= \frac{P(z|s',a)\sum_{s\in S} P(s'|s,a)P(s|b_t)}{P(z|a,b_t)} \\ &= \frac{\mathcal{O}(s',a,z)\sum_{s\in S} \mathcal{T}(s,a,s')b_t(s)}{P(z|a,b_t)} \end{aligned}$$

In essence the above equation evaluates the probability of ending up in state s' given that the agent had a belief about its own state b_t, executed an action a and perceived an observation z according to the predefined observation and transition functions of the POMDP, $\mathcal{O}(\cdot)$ and $\mathcal{T}(\cdot)$ respectively. The denominator $P(z|a,b_t)$, is a normalizing factor and is equal to the total probability of perceiving the observation z given the previous belief state of the agent and the action it executed :

$$
\begin{aligned}
P(z|a, b_t) &= \sum_{s' \in S} P(z|s', a) P(s'|s, a) b_t(s) \\
&= \sum_{s' \in S} \mathcal{O}(z, s', a) \mathcal{T}(s, a, s') b_t(s)
\end{aligned}
$$

2.3 Solving POMDP's

Solving a POMDP amounts to computing an optimal *policy*. A *policy* is a mapping that specifies the action the agent should execute for any possible state that it might occupy. In a POMDP formulation, the true state the agent occupies is never completely known since the agent maintains a belief over all possible states. Therefore, the computed *policy* provides a mapping of belief states to actions.

The optimal action to be executed when the agent occupies a state s_t, is the one with the maximum expected accumulated reward,

$$
E\left[\sum_t \gamma^t \mathcal{R}(s_t, a_t)\right],
$$

where γ is a discount factor that determines how important are the future rewards the robot will receive. If γ is zero, the robot will maximize the reward it will receive for the next time step only. The expected accumulated reward can be computed either for a specific number of steps, the *finite horizon* case, or until the agent reaches the goal state, the *infinite horizon* case.

The function that maps each state of the belief to the corresponding expected accumulated reward is called a *value function*. The t-step optimal value function

[15] is constructed iteratively by *value iteration*. In the case of Markov Decision Processes (MDPs), where the agent's state is fully observable, the t-step optimal value function is formulated as:

$$V_t^*(s) = \max_{a \in \mathcal{A}} \left[\mathcal{R}(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{T}(s,a,s') V_{t-1}(s') \right].$$

However, in POMDPs where the agent's state is partially observable, the value function has to be defined over the whole belief state instead of a single state as in MDPs. Hence, for POMDPs the the t-step optimal value function becomes:

$$V_t^*(b) = \max_{a \in \mathcal{A}} \left[\rho(b,a) + \gamma \sum_{b' \in \mathcal{B}} \tau(b,a,b') V_{t-1}(b') \right],$$

where $\mathcal{B}$ is the set of all possible belief states.

As can be observed in the above equation, the defined transition, $\mathcal{T}(\cdot)$, and reward, $\mathcal{R}(\cdot)$, functions have been replaced by the functions $\tau(\cdot)$ and $\rho(\cdot)$, respectively. This is because the transition and reward functions have to be defined over a belief state, b, instead of a single state, since the true state of the agent is not completely known. Hence, the new functions are defined as:

$$\tau(b,a,b') = P(b'|a,b)$$

and

$$\rho(b,a) = \sum_{s \in S} b(s) R(s,a).$$

The iterative construction of the optimal value function over the set of all

possible belief states $\mathcal{B}$ is an extremely computational expensive procedure. It has been shown that finding an exact solution of a POMDP with infinite horizon is intractable [13]. Therefore, a number of techniques have been proposed for approximating the value function. Many approximation methods are based on solving the underlying fully observable Markov Decision Process (MDP) [25, 1, 12]. More recent approximation methods are those based on state-space compression [21], belief compression [24] and point-based value iteration where the POMDP is solved for a sampled set of belief points [4, 22, 26, 6].

Two very commonly used MDP-based heuristics for approximating the value function are the *most likely state* (MLS) heuristic [25] and the Q_{MDP} approximation [12]. The MLS heuristic solves the underlying MDP for the state with the highest assigned probability. Therefore, the value function becomes:

$$V_t^*(s) = \max_{a \in \mathcal{A}} \left[\mathcal{R}(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{T}(s,a,s') V_{t-1}(s') \right].$$

The Q_{MDP} method solves again the underlying MDP by defining the *Q-functions* as:

$$Q_t(s,a) = \mathcal{R}(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{T}(s,a,s') V_{t-1}(s'),$$

and the optimal value function is then determined by:

$$V_t^*(b) = \max_{a \in \mathcal{A}} \left[\sum_{s \in S} b(s) Q_t(s,a) \right].$$

All of the approximation methods mentioned above have been applied successfully to problems with at most a few thousand states. However, the navigation problem in realistic environments, i.e. the problem considered in this paper, is orders of magnitude larger than these approximation methods can

handle, and hence in this paper a hierarchical representation of POMDPs for robot navigation is proposed.

3 Formulation of POMDPs for the Autonomous Robot Navigation Problem

In the following we present a formulation of POMDPs for autonomous robot navigation in a unified framework. The POMDP decides the actions the robot should perform to reach its goal and also robustly tracks the robot's location in a probabilistic manner. In this paper, we are interested in dynamic environments and hence the POMDP also performs obstacle avoidance. All three functionalities are carried out without the intervention of any other external module.

The elements of the POMDP, $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \mathcal{Z}, \mathcal{O} \rangle$, are instantiated as follows:

set of states, $\mathcal{S}$: Each state in $\mathcal{S}$ corresponds to a discrete entry cell in the environment's occupancy grid map (OGM) and an orientation angle of the robot with respect to a global reference system.

set of actions, $\mathcal{A}$: It consists of all possible rotation actions from 0° to 360° termed as "action angles". The discretization of the robot orientation angles and action angles depends on the number of levels of the POMDP hierarchy (see later Section 4).

set of observations, $\mathcal{Z}$: The observation set is the element of the POMDP that assists in the localization of the robot, that is the belief update after

an action has been taken. The set of observations is instantiated as the output of the *iterative dual correspondence* (IDC) [14] algorithm for scan matching. At each time step, an observation is obtained by feeding the IDC with the current scan of the robot and a reference scan of the environment in which the robot operates. The IDC also requires an estimate of the robot's position from which the current scan was obtained, which is given as the robot's position before it performed the action. This position is taken to be the most likely state of the robot's belief state. It is reasonable to assume that the robot cannot move too far from its previous state at a single time step. Therefore, the output of the IDC algorithm, that is the dx, dy and $d\theta$ from the estimated location provided, will be within certain limits. The output of the IDC algorithm is discretized and thus the set of observations remains small and manageable.

reward function, $\mathcal{R}$: Since the proposed POMDP is used as a unified framework for robot navigation that will provide the actual actions the robot will perform and also carry out local obstacle avoidance for moving objects, the reward function is updated at each time step. The reward function is built and updated at each time step, according to two *reward grid maps* (RGMs): a *static* and a *dynamic* [2]. The RGM is defined as a grid map of the environment in analogy with the OGM. Each of the RGM cells corresponds to a specific area of the environment with the same discretization of the OGM, only that the value associated with each cell in the RGM represents the reward that will be assigned to the robot for ending up in the specific cell. The static RGM is built once by calculating the distance of each cell to the goal position and by incorporating information about

cells belonging to static obstacles. The dynamic RGM is responsible for incorporating into the model information about the current state of the environment, i.e. whether there are objects moving within it or other unmapped objects. In our implementation the robot perceives the environment by taking horizontal laser scans. Hence, at each time step the current laser scan is used to detect the location of objects that are not present in the map. The location of all detected objects form the dynamic RGM where the corresponding cell values are zeroed. Superimposing the static and dynamic RGMs provides the reward function that is updated at each time step. It should be noted that the choice of including in the reward function information about moving objects has alleviated the need of modelling moving objects as observations. Modelling the position of moving objects as observations would increase dramatically the size of observations since it would have to be at least equal to the size of the grid of the modelled environment.

transition and observation functions, $\mathcal{T}$ and $\mathcal{O}$: They are initially defined according to the motion model of the robot and then they are learned as explained in Section 5. Since observations have been defined to depend only on the robot motion when an action is executed, the observation function can also be defined according to the motion model.

4 The Robot Navigation-Hierarchical POMDP (RN-HPOMDP)

HPOMDPs have been recently studied and two approaches have been proposed by Theocharous [27] and Pineau [23, 18]. Our approach to HPOMDP has been developed independently and concurrently with these two approaches and preliminary versions of it have been presented in [2, 3]. POMDP solution methods suffer from the "curse of dimensionality" [7] and also the "curse of history" [4]. Applying both state space and action space hierarchy, as in the RN-HPOMDP, both curses can be harnessed. In the following we present the structure of the RN-HPOMDP along with the methodoly used for learning and planning with the RN-HPOMDP in Sections 5 and 6, respectivelly. A detailed comparison of our approach and the other two approaches present in the literature can be found in Section 7.

4.1 RN-HPOMDP structure

The RN-HPOMDP is built through an automated procedure using as input the map of the environment and the desired discretization of the state and action space. The map of the environment can be either a probabilistic grid map obtained at the desired discretization or a CAD map.

4.1.1 Determining the number of levels of hierarchy of the RN-HPOMDP

The RN-HPOMDP structure is built by decomposing a flat POMDP with large state and action space into multiple POMDPs with significantly smaller state and action spaces. Therefore, in levels other than the bottom level, POMDPs are composed of states and actions that have a coarse discretization and do not represent that actual state the robot occupies or the actual action the robot will perform. Hence they are termed as *abstract states* and *abstract actions* [27]. The process of building the hierarchical structure is performed in a top-down approach. The number of levels of the hierarchical structure is determined by the desired discretization of the action angles or the orientation angles, since their discretization is the same in the RN-HPOMDP structure. The discretization of the orientation and action angles has been chosen to be the same in the RN-HPOMDP structure but this is a designer's choice and is not compromising as it does not affect the performance of the RN-HPOMDP. Thus, if the desired discretization of the action angles or the orientation angles is ϕ, the number of levels of the RN-HPOMDP structure, L, will be $L = \log_2(90^\circ/\phi) + 1$. As explained in the following sections, the top-level of the RN-HPOMDP has a discretization of angles of 90° and at each subsequent level the discretization is doubled. Hence, the number of levels of the hierarchical structure is given by the log_2 of the ratio of the top-level discretization and the desired discretization plus one level that is the top-level.

The number of levels of the RN-HPOMDP structure in conjunction with the desired discretization of the state space affects the size of the top-level POMDP

and in effect the performance of the RN-HPOMDP in time complexity as it will be explained in the following sections. Therefore, the choice of the number of levels of the RN-HPOMDP structure should be made by considering the desired discretization of the state and action space but also the resulting size of the top-level POMDP.

4.1.2 Construction of the top-level of the RN-HPOMDP

The top level of the hierarchical structure is composed of a single POMDP with very coarse resolution. Hence it can represent the whole environment with a small number of abstract states. The grid resolution of the top level states is equal to $d \times 2^{L-1}$, where d is the desired discretization of the whole RN-HPOMDP structure and L is the number of levels of the structure. The orientation angle of the robot and the action angles are also discretized in a very coarse resolution of 90° and thus represent the basic four directions $[0^\circ, 90^\circ, 180^\circ, 270^\circ]$.

The total number of states of the top level POMDP is equal to $|\mathcal{S}^0|/2^{2(L-1)}$, where $|\mathcal{S}^0|$ is the number of states of the corresponding flat POMDP. The number of states of the top level POMDP is reduced once by 2^{L-1} because of the coarser grid resolution and again by 2^{L-1} because of the coarser resolution of the orientation angle, as compared to the corresponding flat POMDP respectively.

To summarize, the top-level is always composed of a single POMDP with predefined discretization of the orientation and action angles at 90°. The state space size of the top-level POMDP is variable and dependent to the discretization of the corresponding flat POMDP and the number of levels of the hierarchical structure. Hence, the number of levels of the HPOMDP structure, L,

should be such that it ensures that the size of the top-level POMDP remains small.

4.1.3 Construction of the intermediate levels of the RN-HPOMDP

Subsequent levels of the HPOMDP are composed of multiple POMDPs, each one representing a small area of the environment and a specific range of orientation angles. The actions of an intermediate level POMDP are a subset of the actions of the corresponding flat POMDP.

In detail, each state of the top level POMDP corresponds to a POMDP at the immediate next level, as we go down the hierarchical structure. A POMDP at an intermediate level l, has states that represent grid locations of the environment at a resolution of $d \times 2^{(L-l)}$, where l is the current intermediate level. Thus, by going down the hierarchical structure the grid resolution of a level's POMDP is twice the resolution of the previous level. Therefore, when a top level state, that corresponds to a specific grid location, is decomposed it will be represented in the immediate next level POMDP by an area of 2×2 cells with double resolution than the top level's resolution.

Orientation angle decomposition

Going down the hierarchical structure, the resolution of the orientation angle is also doubled. Since the resolution of the orientation angle is increased as we go down the hierarchical structure, the whole range of possible orientation angles, $[0°, 360°]$, cannot be represented in every intermediate level POMDP. This would dramatically increase the size of the state space and therefore we

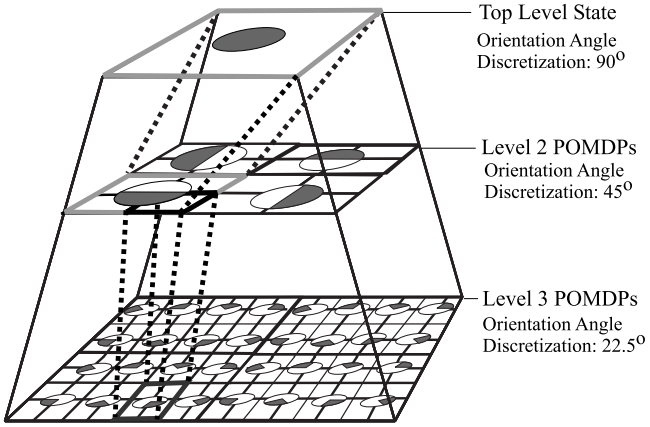

Figure 1: State space hierarchy decomposition. The figure depicts the decomposition of a top level state to lower level states. The top level state corresponds to 4 POMDPs at level 2, each one decomposing the location of the top level state into 4 locations, and its orientation in one of the ranges denoted by the shaded region of the circles for each POMDP. This state decomposition continues at lower levels until the desired discretization of the environment has been reached.

choose to have many POMDPs that represent the same grid location but with a different range of orientation angles.

The range of orientation angles that is represented within each intermediate level POMDP is expressed in terms of the orientation angle, θ_p, of the previous level state that is decomposed, and is equal to

$$\left[\theta_p - \frac{90^\circ}{2^{l-2}}, \theta_p + \frac{90^\circ}{2^{l-2}}\right],$$

where l is the current intermediate level. By the above expression of the range of orientation angles, every intermediate level POMDP will always have five

distinct orientation angles. For example, if the state of the top level POMDP, $l = 1$, has orientation angle $\theta_p = 90°$, the range of orientation angles at the next level, $l = 2$, will be equal to $[0°, 180°]$. As mentioned earlier the angle resolution of the top level is always equal to $90°$ and the next level will have double resolution, i.e. $45°$. Therefore, the range of orientation angles $[0°, 180°]$ will be represented by five distinct orientation angles. As shown in Figure 1, the grid location represented by the top level state is decomposed into four POMDPs, where each one represents a different range of possible orientation angles. Consequently, the size of the state space for every intermediate level POMDP is constant and equal to 20, since it always has five possible orientation angles and it represents a 2×2 area of grid locations. It is easily deduced by the expression that determines the range of orientation angles for an intermediate level POMDP, that a grid location represented by a state will correspond at the next level l to $2^{2(l-1)}$ POMDPs.

Action angle decomposition

Action angles are decomposed from the top level POMDP to the next intermediate level in the same manner as with the orientation angles. The resolution of the action angles at each level is the same as the resolution of the orientation angles. Hence, it is equal to $90°/2^{l-1}$. As a result, a top level state is also decomposed into multiple POMDPs, each one with a different range of orientation angles but also with a different range of action angles. The range of an action set is equal to

$$\left[a_p - \frac{90°}{2^{l-2}}, a_p + \frac{90°}{2^{l-2}}\right],$$

where a_p is the previous level action and l is the current intermediate level. The

action angles set is also always composed by five distinct actions according to the above expression.

4.1.4 Construction of the bottom-level of the RN-HPOMDP

The procedure described in the previous section is used to built all intermediate levels of the hierarchical structure until the bottom level is reached. Bottom level POMDPs' state and action space is discretized at the desired resolution as a flat POMDP would be discretized. The bottom level is composed of multiple POMDPs having the same properties as all other intermediate levels' POMDPs, only that the grid location the bottom level POMDPs represent is overlapping by a region r. Overlapping regions are required to be able to solve the bottom level POMDPs for border location states. Table 1 summarizes the properties of the RN-HPOMDP structure.

4.2 The reference POMDP (rPOMDP)

The RN-HPOMDP described in the previous section, can cope with the computational time requirements but cannot address the memory requirements. A flat POMDP would require to hold a transition matrix of size $(|\mathcal{S}^0|^2 \times |\mathcal{A}^0|)$ and an observation matrix of size $(|\mathcal{S}^0| \times |\mathcal{A}^0| \times |\mathcal{Z}|)$, where $|\mathcal{S}^0|$ and $|\mathcal{A}^0|$ are the size of the state space and action space, respectively, of the flat POMDP. The size of the observation space, $|\mathcal{Z}|$, is the same for the flat POMDP and the RN-HPOMDP since there is no observation space hierarchy.

The RN-HPOMDP structure requires to hold the transition and observation matrices for all the POMDPs at all levels. As can be seen in Table 1, the number

Table 1: Properties of the RN-HPOMDP structure with L levels.

	Top Level	**Intermediate Level l**	**Bottom Level**
No of POMDPs	1	$\lvert\mathcal{A}^{l-1}\rvert \times \lvert\mathcal{S}^{l-1}\rvert$	$\lvert\mathcal{A}^{L-1}\rvert \times \lvert\mathcal{S}^{L-1}\rvert$
Size of $\mathcal{S}$	$\lvert\mathcal{S}^0\rvert/2^{2(L-1)}$	20	$5 \times (2+r)^2$
Range of orientation angles	$[0°, 360°]$	$\left[\theta_p - \frac{90°}{2^{l-1}}, \theta_p + \frac{90°}{2^{l-1}}\right]$	$\left[\theta_p - \frac{90°}{2^{L-1}}, \theta_p + \frac{90°}{2^{L-1}}\right]$
Resolution of orientation angles	$90°$	$90°/2^{l-1}$	$90°/2^{L-1}$
Size of $\mathcal{A}$	4	5	5
Range of action angles	$[0°, 360°]$	$\left[a_p - \frac{90°}{2^{l-2}}, a_p + \frac{90°}{2^{l-2}}\right]$	$\left[a_p - \frac{90°}{2^{L-2}}, a_p + \frac{90°}{2^{L-2}}\right]$
Resolution of action angles	$90°$	$90°/2^{l-1}$	$90°/2^{L-1}$

of POMDPs at each level is large and dependent on the size of action space and state space. Consequently, even thought each POMDP's observation and translation matrix is small, the total memory requirements would be extremely large. The RN-HPOMDP has larger memory requirements than the flat POMDP, although the flat POMDP memory requirements are already very hard to manage for large environments. For this reason, the notion of the *reference* POMDP (rPOMDP) is introduced.

The transition and observation matrices hold probabilities that carry information regarding the motion and sensor uncertainty. In the formulation of the autonomous robot navigation problem with POMDPs, as described in Section 3, transition and observation probabilities for a given action, a, and an observation, z, depend actually only on the relative position and orientation of the robot.

The transition probability of a robot from a state s to a new state s', when it has performed an action a, is only dependent on the action a. Therefore when the robot is executing an action a, the transition probability will be the same for any state s when the resulting state s' is defined relatively to the initial state s.

The probability that the robot observes a feature z, when it is in a state s and performs an action a, can also be defined in the same manner as with the transition probabilities, since the set of features $\mathcal{Z}$ is the result of the scan matching algorithm when feeded with a reference laser scan and the actual scan the robot perceived (cf. Section 3). Therefore, perceived features are dependent on the motion of the robot, i.e. the action a it performed.

The rPOMDP is built by defining a very small state space, defined as an $R \times R$ square grid (in our implementation $R = 7$) representing possible locations of the robot and all the orientation angles of the robot that would be assigned in the flat POMDP. The center location of the state space represents the invariant state s_r of the robot. The action and observation spaces are defined in the same manner they would be defined for the corresponding flat PODMP. This rPOMDP requires to hold transition and observation matrices of size $((R \times 2^{2+L})^2 \times |\mathcal{A}|)$ and $((R \times 2^{2+L})^2 \times |\mathcal{A}| \times |\mathcal{Z}|)$, respectively. The size of the matrices is only dependent on the size of the set of actions and observations and the number of levels of hierarchy, L, since the number of levels defines the discretization of the robot's orientation angle. By the above, it is obvious that no matter how big is the environment that is to be modelled with the RN-HPOMDP the use of the rPOMDP allows to have reasonably sized matrices, depending on the choice made for R, that are easy to maintain and learn.

Given the rPOMDP, transition and observation probabilities for each POMDP in the RN-HPOMDP hierarchical structure are obtained by translating and rotating the reference transition and observation probability distributions over the current POMDP state space, as shown in Figure 2. The transfer of probabilities is performed on-line while a POMDP is solved or the robot's belief is updated.

The transition probability for any POMDP of the hierarchical structure, $\mathcal{T}(s, s', a)$, is equivalent to the transition probability of the rPOMDP, $\mathcal{T}_r(s_r, s'_r, a_r)$. The reference result state, s'_r, is determined by the following equation:

$$\begin{bmatrix} x'_r \\ y'_r \\ f'_r \end{bmatrix} = \begin{bmatrix} x_r \\ y_r \\ f_r \end{bmatrix} + \begin{bmatrix} x' - x \\ y' - y \\ f' - f \end{bmatrix},$$

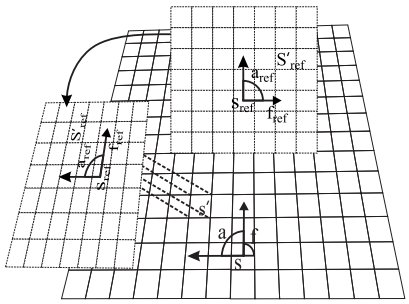

Figure 2: Translation and rotation of the reference POMDP transition probabilities matrix.

where, the states s, s', s_r and s'_r are decomposed to the location and orientation triplets $(x, y, f), (x', y', f'), (x_r, y_r, f_r)$ and (x'_r, y'_r, f'_r), respectively. The reference action is determined by $a_r = a + f - f_r$.

In the same manner, the observation probability for any POMDP of the hierarchical structure, $\mathcal{O}(s, z, a)$, is equivalent to the observation probability of the rPOMDP, $\mathcal{O}_r(s_r, z_r, a_r)$. The reference observation, z'_r, is now determined as:

$$\begin{bmatrix} dx_r \\ dy_r \\ df_r \end{bmatrix} = \begin{bmatrix} d\cos(f_r + a_r) \\ d\sin(f_r + a_r) \\ df \end{bmatrix},$$

where the observations z and z_r are decomposed into (dx, dy, df) and (dx_r, dy_r, df_r), respectively, since observations are defined as the position and angle difference between laser scans, and d is the distance $d = \sqrt{dx^2 + dy^2}$.

5 RN-HPOMDP learning

Since a POMDP is a probabilistic model, learning the parameters of this model, i.e. the transition and observation matrices, is crucial to the performance of the POMDP model and specifically to its performance in keeping track of the robot's true position and orientation.

In our proposed HPOMDP structure, learning is performed only for the *reference* POMDP, since the latter transfers its learned parameters to the whole hierarchical structure, as described in Section 4.2.

Learning the POMDP parameters is performed by initializing the probability matrices and adjusting their parameters iteratively according to an execution trace, that is composed of action and observation pairs, to maximize the likelihood that the execution trace was obtained by the model. The Baum-Welch [10] algorithm is utilized for this purpose.

Since learning is performed only for the rPOMDP, when collecting data for the execution trace the observation and action pairs are converted to *reference* observations and actions. Conversion is performed by the inverse procedure described in Section 4.2. Consequently, learning is performed very fast since the rPOMDP has a very small state space.

5.1 Evaluation of the learned model

In order to test the validity of the learning procedure, we have set up an experiment aiming at a quantitative evaluation of the model that results from a learning session in specific environments. Two learning sessions have been per-

formed; a learning session in a simulated environment where the ground truth is available and also one in a real environment. The environment chosen for both experiments is the FORTH main entrance hall, as shown in Figure 4.

In both experiments, execution traces have been collected where the robot goes from a start state to a goal state. The start and goal states were different for each execution trace. The RN-HPOMDP for both experiments was built by discretizing the environment into $5cm^2$ cells with 5 levels of hierarchy, that results to a discretization step of the orientation and action angles of $5.625°$. The model "appropriateness" has been evaluated using the fitness and entropy measures defined in [10] as:

$$fitness = 1/T \times \ln p(o_{1...T}|a_{1...T})$$

$$entropy = 1/(T \ln |S|) \times \sum_{t=1...T} \sum_{s \in S} [\alpha_t(s) \ln(\alpha_t(s))].$$

Fitness and entropy are indicative measures of how well the model explains an execution trace and how certain the robot is about its position, respectively. The Baum-Welch algorithm is repeated for a number of epochs until it converges. The fitness and entropy measures are graphically shown in Figure 3 for each training epoch. Ideally, fitness and entropy should converge to zero after a sufficiently large number of training epochs. As expected, convergence to zero is not achieved, as its the case with all learning procedures. Still, after a rather small number of epochs, fitness and entropy converge to low values, indicating the validity of the learned model.

In order to provide additional quantitative results on the model accuracy, the position and orientation accuracy in maintaining the robot's state was measured and is shown in Table 2. The peak of the POMDP's belief distribution was used

as the model's estimate of the robot's current state. As can be observed, the figures indicate increased accuracy of the learned model.

In the simulated environment experiments, where the ground truth is available, the position and orientation errors were measured at each time step during execution between start and goal points.

In the real environment experiments, two distinct robot locations were manually marked on the floor of the FORTH main entrance hall, as shown in Figure 4. The robot was driven manually, as accurately as possible, to one of the marked locations and the other marked location was set as the goal position the robot had to reach. The error in the x, y location and orientation between the robot's position after executing the trace obtained by the POMDP model and the marked location it had to reach, was measured manually as accurately as possible.

The mean position and orientation error for both experiments is very close to the discretization of the POMDP, as indicated by the entropy and fitness measures of the learned models. Both experiments, validated that the learned POMDP models were consistent during execution in terms of maintaining the robot's belief and also in reaching the goal position.

6 RN-HPOMDP planning

Solving the RN-HPOMDP to obtain the action the robot should perform, involves solving a POMDP at each level. The intuition of the RN-HPOMDP solution is to obtain at first a coarse path that the robot should follow to reach

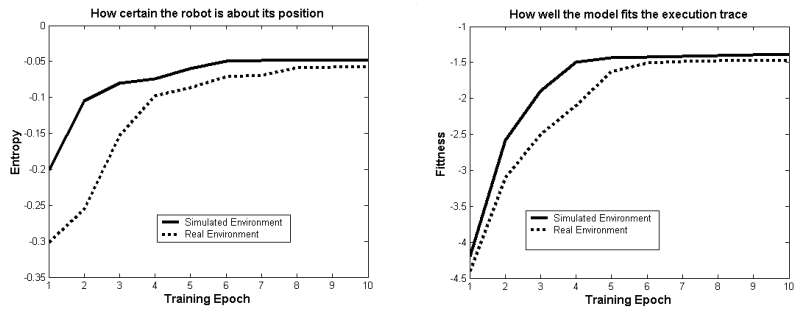

Figure 3: Evaluation of the learned RN-HPOMDP model.

Table 2: Position and orientation accuracy of the learned model.

Mean Error	**Real Environment**	**Simulated Environment**
$x(m)$	0.053	0.023
$y(m)$	0.061	0.041
$f(deg)$	5.525	5.041

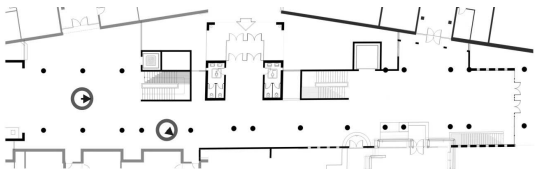

Figure 4: The marked locations in the environment where the experimental evaluation of the RN-HPOMDP model was performed.

a goal position, and then refine this path at each subsequent level in the area that the robot's current position lies. The algorithm that implements the above is presented in Table 3 and its details are explained in the following.

During the RN-HPOMDP planning procedure the belief distribution of the corresponding flat POMDP is maintained at all times. This distribution will be denoted as the *full belief*. Before solving any POMDP at any level, the *full belief* is compressed, by the functions `compressTopBelief()` and `compressBelief()`, to obtain the belief distribution of the POMDP to be solved. Belief compression is performed according to the state abstraction present at each level of the RN-HPOMDP structure, i.e. the discretization reduction of each level as compared to the the discretization of the corresponding flat POMDP. Therefore, the belief assigned to an abstract state, a state with coarse discretization at any level of the hierarchical structure other than the bottom level, will correspond to the average belief of all the corresponding flat POMDP states that the named abstract state has integrated. The belief distribution obtained for any POMDP is normalized before solving it.

The top level POMDP is solved, by the function `solveTopLevel()`, at an infinite horizon, until the goal state is reached. The top level POMDP produces abstract actions, i.e. actions at a coarse resolution that infer only the general direction the robot should follow and not the actual action it will perform. The abstract action to be executed, a_p, as dictated by the top level POMDP solution, determines which POMDP at the immediate next level of the hierarchical structure will be solved to obtain a new refined abstract action, that has a finer discretization but still it is not the actual action the robot will perform.

Table 3: RN-HPOMDP planning

---

while not reached the goal state

`compressTopBelief`$(top\ level)$

$a_p =$ `solveTopLevel`$(top\ level)$

for $l = 2$ to L

$whichPOMDP =$ `selectPOMDP`(l, a_p)

`compressBelief`$(l, whichPOMDP)$

$a_p =$ `solveLevel`$(l, whichPOMDP)$

end

`executeAction`(a_p)

$z =$ `getObservation`$()$

$belief_L =$ `updateBelief`$(whichPOMDP, a_p, z)$

$full\ belief =$ `updateFullBelief`$(belief_L, whichPOMDP)$

end

The POMDP to be solved at the next level is determined by the function `selectPOMDP()`. This function searches a level l for the POMDP among all POMDPs in that level that satisfies the following two criteria:

- The zero moment of the full belief distribution over the area that is defined by the candidate POMDP states is maximum.
- The set of actions of the candidate POMDP contains an action that has minimum distance from the the previous level solution's action, a_p.

The structure of the RN-HPOMDP, as described in Section 4, ensures that when solving an intermediate level POMDP the action obtained from the previous level will be refined to a new action since the action subset range is equal to

$$\left[a_p - \frac{90^\circ}{2^{l-2}}, a_p + \frac{90^\circ}{2^{l-2}}\right].$$

Therefore the solution of an intermediate level POMDP is bounded according to the previous level solution.

The described procedure continues until the bottom level is reached where an abstract action will be refined to an actual action, that is the action the robot will perform.

When the robot executes the action obtained by the bottom level POMDP solution, an observation, z, is obtained and the belief distribution of this bottom level POMDP is updated by `updateBelief`(). Bottom level POMDPs are composed of actual states and actions, i.e. subsets of states and actions that compose the corresponding flat POMDP. Hence, updating the belief of a bottom level POMDP, $belief_L$, amounts to updating a specific region of the *full belief*.

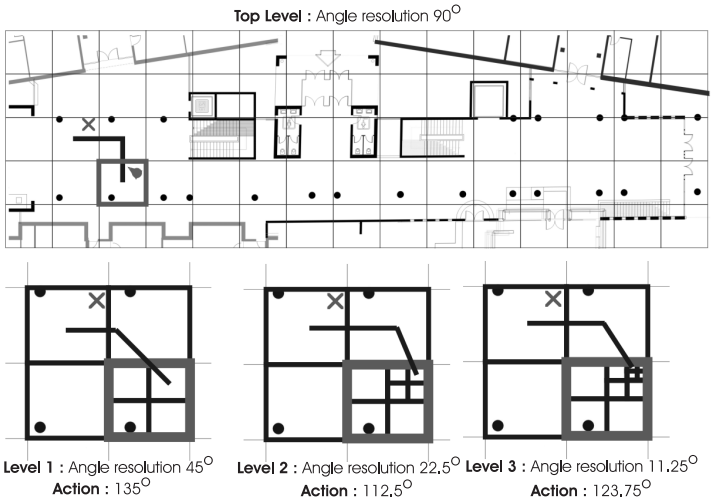

Figure 5: Planning with the RN-HPOMDP.

Therefore, the belief distribution of the bottom level POMDP that was solved is transferred to the *full belief* by the function `updateFullBelief`().

In the current implementation, all POMDPs at all levels are solved using the Voting heuristic (explained in Section 2.3). However, this is not an inherent feature of the RN-HPOMDP structure, as any other POMDP solution method can be used. Furthermore, the POMDP solution method used can also be different for each level of the hierarchical structure.

6.1 Complexity Analysis

In the complexity analysis that follows, execution times are evaluated for the POMDP solution using exact methods and heuristics. The Voting or MLS heuristic, that have the same complexity, will be assumed as the heuristic used

for solving the POMDP. This will assist in the comparison between the RN-HPOMDP and the other hierarchical approaches present in the literature that use the above mentioned heuristics. As already mentioned, in our implementation the Voting heuristic is used to solve POMDPs at all levels.

6.1.1 Approximate Solution

The flat POMDP solution - when solved with the MLS or Voting heuristic - has time complexity for a single step, $\mathbf{O}\left(|\mathcal{S}|^2|\mathcal{A}|\right)$. This makes the POMDP solution intractable when dealing with real world environments at an acceptable resolution, e.g. having more than 10 million states as in our application.

Obtaining a solution by the RN-HPOMDP can dramatically improve the computation time required. Referring to Table 1, where the properties of the RN-HPOMDP structure are detailed, the solution of the top level POMDP requires

$$\mathbf{O}\left(\left(\frac{|\mathcal{S}|}{2^{2(L-1)}}\right)^2 \times 4\right)$$

computational time, where L is the number of levels of the hierarchical structure.

The solution of all intermediate levels POMDPs requires $\mathbf{O}(C_1)$ time, since the size of the state space and action space is constant and predefined. The bottom level POMDP solution is $\mathbf{O}(C_2)$, since the state space and action space is again constant and predefined.

Therefore, the total computational time required to solve the RN-HPOMDP is

$$\mathbf{O}\left(\left(\frac{|\mathcal{S}|}{2^{2(L-1)}}\right)^2\right) + (L-2) \times \mathbf{O}\left(C_1\right) + \mathbf{O}\left(C_2\right),$$

which becomes

$$\mathbf{O}\left(\left(\frac{|\mathcal{S}|}{2^{2(L-1)}}\right)^2\right),$$

that is actually the complexity of the top level POMDP. The top-level POMDP state and action space size can remain small regardless of the size of the whole environment by increasing the number of levels, L, of the hierarchical structure.

6.1.2 Exact Solution

When solving the POMDP exactly for a single step in time t, the time complexity is

$$\mathbf{O}\left(|\mathcal{S}|^2|\mathcal{A}||V_{t-1}|^{|\mathcal{Z}|}\right),$$

where $|V_{t-1}|$ is the number of linear components required to represent the value function at time $t-1$. The size of the value function at any time t is equal to

$$|V_t| = |\mathcal{A}||V_{t-1}|^{|\mathcal{Z}|}.$$

As explained previously, the time complexity of solving the RN-HPOMDP is equal to the time complexity of solving the top level POMDP. The top level POMDP of our hierarchical structure has reduced state space and is equal to $\left(\frac{|\mathcal{S}|}{2^{2(L-1)}}\right)$, where L is the number of levels. Furthermore, the action space is constant and equal to four. Therefore, the time complexity and size of the RN-HPOMDP when solved exactly is

$$\mathbf{O}\left(\left(\frac{|\mathcal{S}|}{2^{2(L-1)}}\right)^2 |V_{t-1}|^{|\mathcal{Z}|}\right)$$

and

$$|V_t| = |V_{t-1}|^{|\mathcal{Z}|},$$

respectively.

Apart from the notable reduction in computation time due to the reduced size of the state and action space, it should be noted that the above mentioned times are for a single time step. The infinite horizon solution of a flat POMDP would require these computations to be repeated for a number N of time steps until the goal point is reached, that is dependent on the number of states of the flat POMDP, $|\mathcal{S}|$. In the RN-HPOMDP case, only the top level POMDP is solved at an infinite horizon, and the number of time steps N' until the goal point is reached, is now dependent on the number of states of the top level POMDP, $(|\mathcal{S}|/2^{2(L-1)})$.

From the above complexity analysis, we may conclude that the proposed approach takes care of the "curse of dimensionality" [7] and also the "curse of history" [4].

7 Comparison with other HPOMDP structures

In this section we compare the RN-HPOMDP against the other two HPOMDP approaches present in the literature, in terms of time complexity for solving the HPOMDP, state space and action space abstraction methodology, and the application framework of the HPOMDP. Finally, a comparison of the RN-HPOMDP and the most recent approximation methods for solving a flat POMDP is presented.

7.1 Comparison with the Theocharous approach

The Theocharous [27] approach uses a topological map of the environment where the state abstraction in high levels of the HPOMDP, has a physical meaning based on the environment. Thus, abstract states are manually defined such that they represent a corridor or a junction. On the other hand, the RN-HPOMDP is built through an automated procedure that requires as input only a probabilistic occupancy grid map or a CAD map of the environment.

The Theocharous HPOMDP has been used as a high-level planner where the POMDP is solved once to obtain the shortest path to the goal position. As a result, the state space resolution is set to $2m^2$ and the action space is discritized at a resolution of 90°. Our approach models the environment at a fine resolution (e.g. $5cm^2$) and the action resolution can be discretized up to 1° based on the number of levels of hierarchy. Finally, the RN-HPOMDP is used as a global planner that is solved at each time step to provide the actual actions the robot will perform without the intervention of any other intermediate modules. The RN-HPOMDP integrates the modules for planning, localization and local obstacle avoidance.

The Theocharous approach, uses the MLS heuristic and has time complexity between $\mathbf{O}(|\mathcal{S}|^{\frac{2}{d}}N|\mathcal{A}|)^{(see\ 2)}$ and $\mathbf{O}(|\mathcal{S}|^2|\mathcal{A}|)^{(see 3)}$, based on how well the HPOMDP was constructed. The time required to solve the RN-HPOMDP is

[2] d is the depth of the tree and N is the maximum number of entry states for an abstract state.

[3] The size of the action space $|\mathcal{A}|$ was added in the time complexity of the Theocharous approach, so that the comparison with the complexity of a flat POMDP and our approach can be direct, although in their approach $|\mathcal{A}|$ is constant and equal to 4.

$\mathbf{O}((|\mathcal{S}|/2^{2(L-1)})^2)$, hence the complexity reduction of our approach is significantly greater and also is not dependent on any quality measure of the hierarchical structure.

7.2 Comparison with the Pineau approach

In the Pineau HPOMDP approach [18], actions are grouped into abstract actions called subtasks. Subtasks are defined manually and according to them state abstraction is performed automatically. States that have the same reward value for executing any action that belongs to a predefined subtask are clustered. Observation abstraction is performed by eliminating the observations that have zero probability over all state clusters for that actions belonging to a specific subtask.

Planning with the Pineau HPOMDP involves solving the POMDP defined for each action subtask. POMDPs are solved using the exact POMDP solution method.

The HPOMDP proposed by Pineau does not have a guaranteed reduction of the action space and state space since it is dependent on the action abstraction that is defined manually. The authors have performed experiments (real and simulated) only for problems of high level behavior control. Hence it is not clear whether their approach of state abstraction could be applied to the problem of autonomous robot navigation in the context that we have defined or, more importantly, if it would perform as efficiently as our approach does, since the latter has a guaranteed reduction of the state space that is equal to $\left(|\mathcal{S}|/2^{2(L-1)}\right)$. On the other hand, the authors in [18] do not state how well

their approach performs in terms of state space abstraction.

The HPOMDP proposed by Pineau has been used in a real world application for high level robot control and dialogue management [18]. This application has been modelled using 576 states, 18 observations and 19 actions categorized into 3 subtasks. The problems encountered with our approach are many orders of magnitude larger (e.g. $|\mathcal{S}| = 18,411,520$, $|\mathcal{A}| = 256$, $|\mathcal{Z}| = 24$) and can be solved in real time.

7.3 Approximation methods for solving flat POMDPs

A short discussion on the performance of approximation methods for solving flat POMDPs follows in this section. This discussion will allow us to elaborate further on the performance of the RN-HPODMP and also necessitate further the need of the proposed hierarchical structure, at least when considering the autonomous robot navigation problem.

In [6] a review of approximation methods for solving POMDPs is presented. The complexity of the methods reviewed can be seen in Table 4. Furthermore, one of the most recent methods for approximation is the Point Based Value Iteration (PBVI) [4] method. The time complexity of PBVI is $\mathbf{O}(|\mathcal{S}||\mathcal{A}||V_{t-1}||\mathcal{Z}||B|)$, where $|B|$ is the size of the finite set of belief points and $|V|$ remains constant throughout iterations.

To summarize, the time complexity of all approximation methods is in the best case polynomial to the size of the POMDP. All the above mentioned methods have been applied to problems where the POMDP was comprised of a few

Table 4: Complexity of solving a POMDP with the approximation methods reviewed in [6].

Approximation Method	**Complexity**
MDP	$\mathbf{O}(\lvert\mathcal{A}\rvert\lvert\mathcal{S}\rvert^2)$
QMDP	$\mathbf{O}(\lvert\mathcal{A}\rvert\lvert\mathcal{S}\rvert^2)$
Fast Informed Bound (FIBM)	$\mathbf{O}(\lvert\mathcal{A}\rvert^2\lvert\mathcal{S}\rvert^2\lvert\mathcal{Z}\rvert^2)$
UMDP	$\mathbf{O}(\lvert\mathcal{A}\rvert\lvert\mathcal{S}\rvert^2\lvert V\rvert_{t-1})$, $\lvert V_t\rvert = \lvert V_{t-1}\rvert\lvert\mathcal{A}\rvert$
Grid based interpolation extrapolation [a]	$\mathbf{O}(\lvert G\rvert\lvert\mathcal{A}\rvert\lvert\mathcal{S}\rvert^2\lvert\mathcal{Z}\rvert C_{eval})$

[a] $|G|$ is the size of a finite set of grid points used to update the value updates and C_{eval} is the computational cost of evaluating the interpolation-extrapolation rule for $|G|$ points, where in some cases this cost can be eliminated.

thousand states. The problem we consider consists of many orders of magnitude larger state space. As a result, the reduction of the state space that the RN-HPOMDP offers and also the reduction of the action space is crucial to its performance. Furthermore, since the proposed hierarchical structure is not restricted to a specific method for solving the underlying POMDPs, a combination of an approximation method for solving a flat POMDP with the proposed hierarchical structure can dramatically improve its performance.

7.4 Computational time comparison

Further to the theoretical comparison presented in the previous section, for indicative comparison purposes we provide the CPU times required to solve the RN-HPOMDP and also the Theocharous and Pineau HPOMDP approaches in Tables 5, 6 and 7. It should be stressed out, that the times referring to the

Table 5: Computation time required to solve a HPOMDP with the compared approaches.

	POMDP size		**CPU time (sec)**
Theocharous [27]	$\|S\| = 575$	$\|A\| = 4$	2.11 - 5.7
	$\|S\| = 1385$	$\|A\| = 4$	5.05 - 26.12
Pineau et. al. [23]	$\|S\| = 11$	$\|A\| = 6$	2.84
	$\|S\| = 20$	$\|A\| = 30$	77.99

Pineau approach are the ones from their initial version of HPOMDP [23] where there was only action space hierarchy. It should be also noted that the CPU times mentioned are the ones the authors state and have not been obtained using computers of the same power. Another point is that the Theocharous approach is solved using the MLS heuristic and in our approach the POMDPs are solved using the Voting heuristic that has the same computational complexity with the MLS heuristic. However, the Pineau HPOMDP is solved using exact methods. Regardless of the mentioned differences, the superior computational performance of our approach can be easily extracted from the tabulated results since the size of the problem is many orders of magnitude larger.

8 Results

The RN-HPOMDP has been tested extensively in a real world environment. The robot was set to operate for more than 70 hours in the FORTH main entrance hall shown in Figure 6. The environment was modeled with a RN-HPOMDP

Table 6: Computation time required to solve the RN-HPOMDP with varying grid size and 5 levels.

Grid size	**POMDP size**		**CPU time (sec)**
$5cm \times 5cm$	$\|S\| = 18,411,520$	$\|A\| = 64$	18.520
$10cm \times 10cm$	$\|S\| = 4,602,880$	$\|A\| = 64$	0.911
$15cm \times 15cm$	$\|S\| = 2,038,080$	$\|A\| = 64$	0.426
$20cm \times 20cm$	$\|S\| = 1,150,720$	$\|A\| = 64$	0.257
$25cm \times 25cm$	$\|S\| = 734,976$	$\|A\| = 64$	0.262
$30cm \times 30cm$	$\|S\| = 503,808$	$\|A\| = 64$	0.251

Table 7: Computation time required to solve the RN-HPOMDP with varying number of levels and grid size of $10cm \times 10cm$.

No. of Levels	**POMDP size**		**CPU time (sec)**
3	$\|S\| = 1,150,720$	$\|A\| = 16$	201.210
4	$\|S\| = 2,301,440$	$\|A\| = 32$	16.986
5	$\|S\| = 4,602,880$	$\|A\| = 64$	0.911
6	$\|S\| = 9,205,760$	$\|A\| = 128$	0.460
7	$\|S\| = 18,411,520$	$\|A\| = 256$	0.411

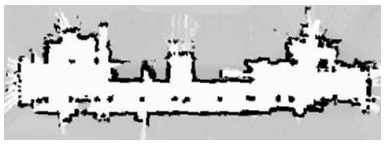

Figure 6: The FORTH main entrance hall.

of size $|S| = 18,411,520$, $|A| = 256$ and $|Z| = 24$. The RN-HPOMDP was built with 7 levels. Experiments were performed in a dynamic environment where people were moving within it. In all cases the proposed navigation model has shown a robust behavior in reaching the assigned goal points and avoiding humans or other objects. A sample path the robot followed to reach its goal and also performed local obstacle avoidance to avoid a human is shown in Figure 7.

(a) (b)

Figure 7: Avoiding a human to reach the goal position. The robot track is marked with the black dots (•) and the human track is marked with the grey dots (•).

9 Conclusions and Future Work

In this work we introduced a new approach to hierarchical POMDPs (HPOMDPs). The proposed approach is designed specifically for the autonomous robot navigation problem, hence termed as Robot Navigation-HPOMDP (RN-HPOMDP). The RN-HPOMDP is utilized as a unified model that caters for planning, localization and local obstacle avoidance. Hence, it is formulated in such a manner that it does not depend on any other external modules for localization and local obstacle avoidance. To the best of our knowledge, it is the first time a POMDP is used to provide the actual actions the robot executes and not as a high level mission planner. The RN-HPOMDP offers significant state space and action space reduction compared to other hierarchical approaches present in the literature. Furthermore, the state space and action space reduction is guaranteed and not dependent on the environment where the robot operates. Additionally, the RN-HPOMDP can be used in conjunction with any approximation method for solving flat POMDPs, to further improve its performance. A novel approach has been also proposed for storing the model parameters with the reference POMDP (rPOMDP). The RN-HPOMDP has been experimentally validated in a real world environment.

Future work involves integrating into the RN-HPOMDP prediction about the motion of humans and other obstacles to perform efficient and effective obstacle avoidance in a predictive manner [2, 3]. Furthermore, the application of the RN-HPOMDP to multi-robot navigation and cooperation will be examined.

Bibliography

[1] Anthony R. Cassandra, Leslie Pack Kaelbling, and James A. Kurien. Acting under uncertainty: Discrete bayesian models for mobile-robot navigation. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1996.

[2] A. Foka and P. Trahanias. Predictive autonomous robot navigation. In *Proceedings IEEE/RSJ International Conference Intelligent Robots and Systems (IROS)*, 2002.

[3] A. Foka and P. Trahanias. Predictive control of robot velocity to avoid obstacles in dynamic environments. In *Proceedings IEEE/RSJ International Conference Intelligent Robots and Systems (IROS)*, 2003.

[4] G. Gordon J. Pineau and S. Thrun. Point-based value iteration: An anytime algorithm for pomdps. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2003.

[5] Milos Hauskrecht. *Planning and Control in Stochastic Domains with Imperfect Information.* PhD thesis, MIT, 1997.

[6] M. Hauskrecht. Value function approximations for partially observable Markov decision processes. *Journal of Artificial Intelligence Research*, 13:33–95, 2000.

[7] Leslie Pack Kaebling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1–2):99–134, 1998.

[8] Ashraf A. Kassim, and B. V. K. Vijaya Kumar. Path planners based on the wave expansion neural network. *Robotics and Autonomous Systems*, 26(1):1–22, 1999.

[9] Oussama Khatib, Sean Quinlan and David Williams. Robot planning and control. *Robotics and Autonomous Systems*, 21(3):249–261, 1997.

[10] Sven Koenig and Reid G. Simmons. Unsupervised learning of probabilistic models for robot navigation. In *Proceedings of the International Conference on Robotics and Automation*, pages 2301–2308, 1996.

[11] Jean-Claude Latombe. *Robot Motion Planning.* Kluwer Academic Publishers, 1991.

[12] Michael Littman, Anthony Cassandra, and Leslie Kaebling. Learning policies for partially observable environments: Scaling up. In *Proceeding of the 12th International Conference on Machine Learning*, pages 362–370, 1995.

[13] Michael L. Littman, Judy Goldsmith, and Martin Mundhenk. The computational complexity of probabilistic planning. *Journal of Artificial Intelligence Research*, 9:1–36, 1998.

[14] F. Lu and E. Milios. Robot pose estimation in unknown environments by matching 2d range scans. *Journal of Intelligent and Robotic Systems*, 18:249–275, 1998.

[15] G.E. Monahan. A survey of partially observable markov decision processes: Theory, models, and algorithms. *Management Science*, 28:1–16, 1982.

[16] U. Nehmzow and C. Owen. Robot navigation in the real world:Experiments with Manchesters FortyTwo in unmodified, large environments. *Robotics and Autonomous Systems*, 33(4):223–242, 2000.

[17] Daniel Nikovski and Illah Nourbakhsh. Learning probabilistic models for decision-theoretic navigation of mobile robots. In *Proc. 17th International Conf. on Machine Learning*, pages 671–678. Morgan Kaufmann, San Francisco, CA, 2000.

[18] J. Pineau and S. Thrun. An integrated approach to hierarchy and abstraction for POMDPs. Technical Report (CMU-RI-TR-02-21), Carnegie Mellon University, 2002.

[19] J. Pineau, G. Gordon and S. Thrun. Applying metric-trees to belief-point POMDPs. In *Neural Information Processing Systems (NIPS)*, 2003.

[20] J. Pineau, M. Montemerlo, M. Pollack, N. Roy and S. Thrun. Towards robotic assistants in nursing homes: challenges and results. *Robotics and Autonomous Systems*, 42(3-4):271–281, 2003.

[21] P. Poupart and C. Boutilier. Value-directed compression of POMDPs. In *Neural Information Systems (NIPS)*, 2003.

[22] K.-M. Poon. *A fast heuristic algorithm for decision-theoretic planning.* Master's Thesis, The Hong-Kong University of Science and Technology, 2001.

[23] N. Roy J. Pineau and S. Thrun. A hierarchical approach to pomdp planning and execution. *Workshop on Hierarchy and Memory in Reinforcement Learning (ICML)*, 2001.

[24] N. Roy and G. Gordon. Exponential family PCA for belief compression in POMDPs. In *Neural Information Systems (NIPS)*, 2003.

[25] R. Simmons and Sven Koenig. Probabilistic robot navigation in partially observable environments. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1080-1087, 1995.

[26] M.T.J. Spaan and N. Vlassis. A point-based POMDP algorithm for robot planning. In *Proceedings of 2004 IEEE International Conference on Robotics and Automation (ICRA)*, 2004.

[27] G. Theocharous. *Hierarchical Learning and Planning in Partially observable Markov Decision Processes.* PhD thesis, Michigan State University, 2002.

[28] S. Thrun, M. Beetz, M. Bennewitz, W. Burgard, A.B. Cremers, F. Dellaert, D. Fox, D. Hahnel, C. Rosenberg, N. Roy, J. Schulte, and D. Schulz. Probabilistic algorithms and the interactive musuem tour-guide robot minerva. *International Journal of Robotics Research*, 19(11):972–999, 2000.