Learning Hierarchical Partially Observable Markov Decision Process Models for Robot Navigation

Georgios Theocharous
theochar@cse.msu.edu

Khashayar Rohanimanesh
khash@cse.msu.edu

Sridhar Mahadevan
mahadeva@cse.msu.edu

Department of Computer Science and Engineering
Michigan State University
East Lansing, MI 48823

Abstract— **We propose and investigate a general framework for hierarchical modeling of partially observable environments, such as office buildings, using Hierarchical Hidden Markov Models (HHMMs). Our main goal is to explore hierarchical modeling as a basis for designing more efficient methods for model construction and useage. As a case study we focus on indoor robot navigation and show how this framework can be used to learn a hierarchy of models of the environment at different levels of spatial abstraction. We introduce the idea of model *reuse* that can be used to combine already learned models into a larger model. We describe an extension of the HHMM model to includes actions, which we call hierarchical POMDPs, and describe a modified hierarchical Baum-Welch algorithm to learn these models. We train different families of hierarchical models for a simulated and a real world corridor environment and compare them with the standard "flat" representation of the same environment. We show that the hierarchical POMDP approach, combined with model reuse, allows learning hierarchical models that fit the data better and train faster than flat models.**

I. Introduction

Recent work in artificial intelligence (AI) on sequential decision-making under uncertainty has adopted the framework of Partially Observable Markov Decision Processes (POMDP) [1], [2]. This framework allows modeling environments where the underlying states are "hidden", and only partially observable through noisy sensory observations and actions. POMDPs extend the well-known Hidden Markov Model (HMM) [3] to include actions and rewards. There are well-developed algorithms for both building and using POMDP models, such as the Baum-Welch procedure and dynamic programming. However, past work on POMDPs has been restricted to "flat" uniform scale models, and the model-learning and planning algorithms scale poorly with the size of the model. Several studies have shown that the HMM/POMDP framework can be used to program autonomous mobile robots to navigate in real office environments [1], [4], [2], but these systems use flat models. As the size of the environment grows, it becomes increasingly difficult to learn and use flat models, and it would be desirable to have a natural way of reusing previously learned sub-models.

In this paper, we address these limitations of earlier standard HMM/POMDP systems using the Hierarchical Hidden Markov Model (HHMM) framework [5]. Using the HHMM approach, a robot learns and uses a hierarchy of homogeneous representations of the environment in which each layer of the hierarchy maintains a probabilistic model of the environment defined at some resolution. The tree structure of HHMM models also provides a natural approach for rapid model learning by reusing previously learned sub-models. We extend HHMM models to include primitive actions, and show a procedure for converting hierarchical models into flat POMDPs.

Intuitively, hierarchical modeling should allow us to learn the environment in a modular fashion and therefore be able to learn faster, and even be able to discover relationships at abstract levels that would not be trivial in flat representations. Fortunately, many natural environments can be viewed at different levels of spatial abstraction. For example, an office environment modeled at a coarse level would consist of nodes for corridors, intersections and dead-ends; at a finer level, each model element would represent a fixed-length region of a corridor.

In this paper we demonstrate through experiments that the HHMM framework extended to the hierarchical POMDP case appears to have some compelling advantages as a basis for designing scalable spatial learning algorithms. We will provide empirical results from learning in both real-world and simulated robot navigation to illustrate the potential of the hierarchical POMDP framework.

II. Hierarchical Hidden Markov Models

The Hierarchical Hidden Markov Model (HHMM) [5] generalizes the standard hidden Markov model (HMM) [3], by allowing hidden states to represent stochastic processes themselves. An HHMM is visualized as a tree structure in which there are three types of states, production states (leaves of the tree) which emit observations, and internal states which are (unobservable) hidden states that represent entire stochastic processes. Each production state is associated with an observation vector which maintains distribution functions for each observation defined for the model. Each internal state is associated with a horizontal transition matrix, and a vertical transition vector. The horizontal transition matrix of an internal state defines the transition probabilities among its children. The vertical transition vectors define the probability of an internal state to activate any of its children. Each internal state is also associated with a child called an *end-state* which returns control to its parent. The end-states do not produce observations and cannot be activated through a vertical transition from their parent. The HHMM is formally defined as a 5 tuple $\langle S, T, \Pi, Z, O\rangle$:

- S denotes the set of states. The functions $p(s)$ denotes the parent of state s. The function $c(s, j)$ returns the j^{th} child of state s. The end-state child of an abstract state s is denoted by e^s. The set of children of a state s is denoted by C^s and the number of children by $|C^s|$. There are three types of states.
  – *Production states*
  – *Abstract states*
  – *End-states*
- $T^s : C^s \times C^s \rightarrow (0, 1)$ denotes the horizontal transition functions, defined separately for each abstract state. A horizontal transition function maps each child state of s into a probability distribution over the children states of s. We write $T^s(c(s, i), c(s, j))$ to denote the horizontal transition probability from the i^{th} to the j^{th} child of state s. As an example, in Figure 1, $T^{s4}(s7, s8) = 0.8$.
- $\Pi^s : \{C^s - e^s\} \rightarrow (0, 1)$ denotes the vertical transition function for each abstract state s. This function defines the initial distribution over the children states of state s, except from the end-state child e^s. For example, in Figure 1, $\Pi^{s4}(s6) = 0.5$.
- Z denotes the set of observations.
- $O : S^{product} \rightarrow (0, 1)$ denotes a function that maps every product state to a distribution over the observation set. We write $O(s, z)$ for the probability of observing z in state s.

Figure 1 shows a graphical representation of an example HHMM. The HHMM produces observations as follows:

1. If the current node is the root, then it chooses to activate one of its children according to the vertical transition vector from the root to its children.
2. If the child activated is a product state, it produces an observation according to an observation probability output vector. It then transitions to another state within the same level. If the state reached after the transition is the end-state, then control is returned to the parent of the end-state.
3. If the child is an abstract state then it chooses to activate one of its children. The abstract state waits until control is returned to it from its child end-state. Then it transitions to another state within the same level. If the resulting transition is to the end-state then control is returned to the parent of the abstract state.

Fine et al. [5] describe a hierarchical Baum-Welch algorithm that is able to re-estimate the model parameters λ (including transitions matrices, vertical vectors, and observation vectors) of an HHMM given observation sequences $M = z_1, z_2, ..., z_T$. We have extended this algorithm to be able to learn the parameters of an HHMM that includes actions, as we describe next.

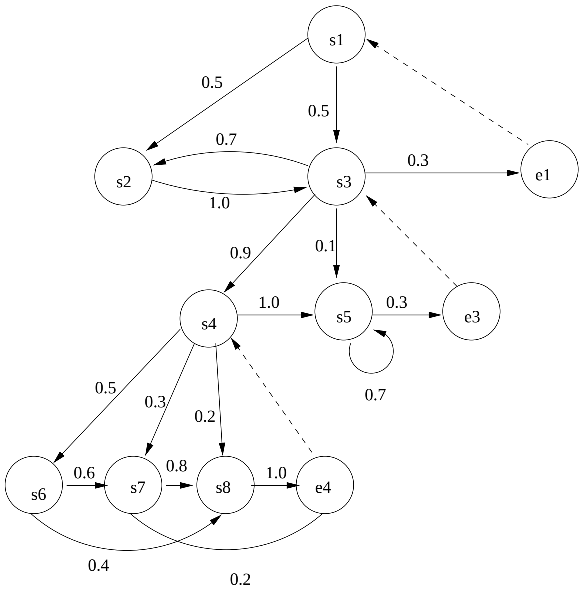

Fig. 1. An example hierarchical HMM. Only leaf (production) states (s2, s5, s6, s7, and s8) have associated observations.

III. Hierarchical Partially Observable Markov Decision Process Models

A. Model definition

Robot navigation is an example of a planning problem that requires extending the HHMM model to include both primitive and abstract actions, as well as reward functions for specifying goals. We call this extended model *hierarchical POMDP* and we formally define it as follows:

- S denotes the set of states which are exactly the same as an HHMM.
- A denotes the set of primitive actions. The primitive actions initiate and terminate in product states. The product states represent the lowest resolution of a physical environment and therefore every action has to start in some product state and end in some other product state. For example, in an indoor robot navigation environment, if abstract states are corridors and product states are 2 meter locations in the corridors, then every primitive action such as "go-forward" will always take the robot from one product state to another product state. However, the transition probability from one product state $s1$ to some other product state $s2$ is not simply a lookup operation into a global transition matrix as in flat POMDPs. In the hierarchical POMDP model we may have to look at more than a single horizontal transition matrix and even multiple vertical vectors. An example of this calculation is shown in Figure 2.
- $T^{s,a} : C^s \times C^s \rightarrow (0, 1)$ denotes the horizontal transition functions, which are defined separately for each abstract state s and action a. A horizontal transition function maps each child state of s and action pair into a probability distribution over the children states of s.
- $\Pi^s : \{C^s - e^s\} \rightarrow (0, 1)$ denotes the vertical transition function for each abstract state s. This function defines the initial distribution over the children states of state s.

- Z denote the set of observations as before.
- $O : S^{product} \to (0,1)$ is a function that maps every product state to a distribution over the observation set.
- $R : S^{product} \times A \to \Re$ denotes an immediate reward function defined on the product states.

Figure 2 shows an example hierarchical POMDP. A hierarchical Baum-Welch algorithm for the hierarchical POMDP can be defined by extending the hierarchical Baum-Welch algorithm for HHMMs. First we define the α variable as shown in Equation 1

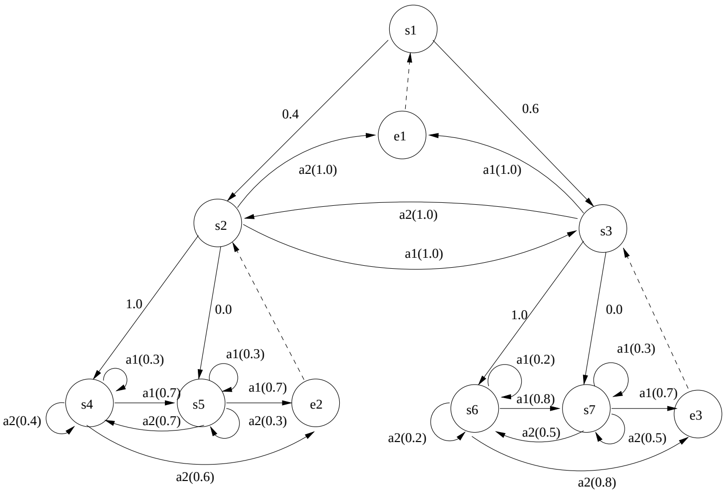

Fig. 2. An example hierarchical POMDP with two primitive actions, $a1$ and $a2$. To calculate the transition from state $s4$ to state $s4$ under action $a2$ we have to consider all non-zero probability paths. One path is $(s4, s4)$ and the other path is $(s4, e2, s2, e1, s1, s2, s4)$. Thus while $T^{s_2,a_2}(s_4, s_4) = 0.4$, the probability of transitioning from s_4 to s_4 under action $a_2 = 0.4 + 0.6 \times 1.0 \times 0.4 \times 1.0 = 0.64$.

$$\alpha(t, t+k, s) = P(z_t, ... z_{t+k}, s\ finished\ at\ t+k\ |a_t, ... a_{t+k-1}, p(s)\ started\ at\ t, \lambda) \tag{1}$$

The α variables gives us the probability of the observation sequence $z_t ... z_{t+k}$ and that state s finished at time $t+k$ given that actions $a_t, ... a_{t+k-1}$ were taken and the parent of s, $p(s)$ was started at time t. We say that a product state finishes at time t after the production of observation z_t and an abstract state finishes when control is returned to it from its child end-state after one of its children produces its last observation z_t. We also say that a product state s, started at time t, if at time t it produced observation z_t. An abstract state s starts at time t if at time t one of its children produced observation z_t but observation z_{t-1} was generated before s was activated by its parent or any other horizontal transition.

Note that using the α variable we can calculate the probability of an observation sequence given the model as shown in Equation 2.

$$P(M|\lambda) = \sum_{s=1|c(root,s) \neq e^{root}}^{|C^{root}|} \alpha(1, T, c(root, s)) \tag{2}$$

The backward variable β, defined in Equation 3, denotes the probability that a state s was entered at time t and that the observations $z_t ... z_{t+k}$ were produced by the parent of s, which is $p(s)$ and actions $a_t, ... a_{t+k}$ were taken and p(s) terminated at time $t+k$.

$$\beta(t, t+k, s) = P(z_t, ... z_{t+k}|a_t, ... a_{t+k},\ s\ started\ at\ t,\ p(s)\ generated\ z_t ... z_{t+k}\ and\ finished\ at\ t+k, \lambda) \tag{3}$$

The next important variable is ξ, defined in Equation 4, which denotes the probability of making a horizontal transition from s to s' at time t.

$$\xi(t, s, s') = P(s\ finished\ at\ time\ t,\ s'\ started\ at\ time\ t+1\ |a_1, ... a_T, z_1, ... z_T, \lambda) \tag{4}$$

Another important variable is $\chi(t, s)$, shown in Equation 5, which defines the probability that state s was activated by its parent $p(s)$ at time t given the action and observation sequence

$$\chi(t, s) = P(p(s)\ started\ at\ time\ t, s\ started\ at\ time\ t|\ a_1, ... a_T, z_1, ... z_T, \lambda) \tag{5}$$

Based on these variables we can re-estimate the model parameters. The vertical vectors are re-estimated in Equations 6 and 7.

$$\Pi^s(n) = \chi(0, n),\ if\ s = root,\ n \in C^s \tag{6}$$

$$\Pi^s(n) = \frac{\sum_{t=1}^{T} \chi(t, n)}{\sum_{i=1|c(s,i) \neq e^s}^{|C^s|} \sum_{t=1}^{T} \chi(t, c(s, i))},\ if\ s \neq root,\ n \in C^s \tag{7}$$

The horizontal transition matrices are re-estimated in Equation 8. The re-estimation calculates the average number of times the process went from state s to state s' over the number of times the process exited state s under action a.

$$T^{p(s),a}(s, s') = \frac{\sum_{t=1|a_t=a}^{T} \xi(t, s, s')}{\sum_{t=1|a_t=a}^{T} \sum_{i=1}^{|C^{p(s)}|} \xi(t, s, c(p(s), i))} \tag{8}$$

The observation vectors are re-estimated in Equation 9. The re-estimation of the observation model calculates the average number of times the process was in state s and perceived observation z over the number of times the process was in state s.

$$O(s,z) = \frac{\sum_{t=1,z_t=z}^{T} \chi(t,s) + \sum_{t=2,z_t=z} \gamma_{in}(t,s)}{\sum_{t=1}^{T} \chi(t,s) + \sum_{t=2}^{T} \gamma_{in}(t,s)}$$

$$where\ \gamma_{in}(t,s) = \sum_{i=1|c(p(s),i)\neq e^{p(s)}}^{|C^{p(s)}|} \xi(t-1, c(p(s),i), s) \tag{9}$$

B. Planning using Hierarchical POMDPs

Solving a POMDP means that we have to find a mapping from each "belief state" (probability distribution over states) to actions that will achieve the best long term sum of rewards [6]. The belief state is a sufficient statistic that summarizes the past history of observations and actions. There is an efficient Bayesian update procedure that can be used to calculate the belief state probability distribution over all the product states given a sequence of observations and actions. Unfortunately the number of belief states is infinite and exact solutions to large POMDPs are computationally infeasible. Fortunately, many heuristics solutions such as the most likely state heuristic (MLS) and the Q-MDP method are known to provide satisfactory approximate solutions for robot navigation [7]. We have extended these approximate methods to hierarchical POMDPs, by implementing hierarchical versions of these methods (e.g. for the MLS procedure, we compute the most likely abstract state, and then recursively the most likely product state).

Any hierarchical POMDP model with actions can be converted into an equivalent flat POMDP. The states of the equivalent flat POMDP are the product states of the hierarchical POMDP and are associated with a global transition matrix that is calculated from the vertical and horizontal transition matrices of the hierarchical POMDP. To construct this global transition matrix of the equivalent flat POMDP, for every pair of states (s_1, s_2), we need to sum up the probabilities of all the paths that will transition the system from $s1$ to $s2$ under some action a. One such example is shown in Figure 2. However, "flattening" a hierarchical model in this manner will destroy the ability to learn faster by reusing sub-models, and negate one of the primary advantages of the hierarchical approach.

IV. Learning hierarchical POMDPs for robot navigation

We now describe a detailed set of experiments comparing hierarchical POMDP models with flat models in terms of learning speed and fit to the data. We have conducted experiments using both a real robot platform (a Nomad 200 robot) and a simulated environment. Using the Nomad 200 simulator we constructed a model of the second floor of the MSU Engineering building, shown in Figure 3. We also used a real indoor environment, shown in Figure 10.

Such topological maps can be automatically compiled into a Markov representation, either as a flat POMDP or as a hierarchical POMDP model. In our experiments, we used both "good" initial models as well as uninformed weak "ergodic" initial models. Figures 4, 5, and 6 show hierarchical and flat POMDP models. In a good initial model, we provide a priori the appropriate connectivity of the actions and we also initialize the observation models according to the location of a state. An observation consists of 5 components: the action taken and the probabilities that the robot has seen a wall or an opening on its four sides, front, left, back and right. The probabilities of the observations are computed by neural nets which take as input local occupancy grid maps around the robot (constructed from 16 sonars) and output the probability of a wall and opening for every direction [8]. For example, if a product state is facing in the corridor direction, then the initial observation model would be front (wall:0.4, opening:0.6), left (wall:0.6, opening:0.4), back (wall:0.4, opening:0.6), and right (wall:0.6, opening:0.4). In the case of uninformed ergodic initial models, we also provide good observation models, but every state is connected with every other state under the same parent for every action at all levels.

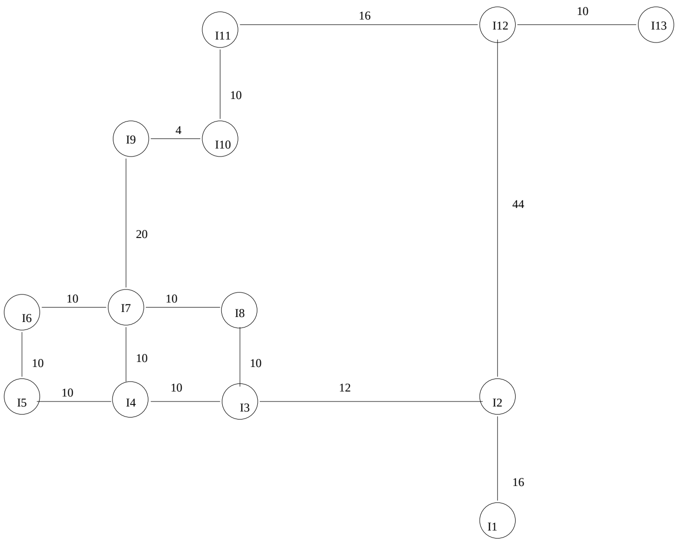

Fig. 3. A topological map of the 2nd floor of the Engineering building. Numbers indicate distances in meters. For each edge and each vertex there are four states at the second level of the hierarchical POMDP. States representing the (North-South or East-West) direction of corridors are expanded to a number of third level product states, each one representing two meters.

In the first experiment, we compared hierarchical and flat POMDPs, where we biased the training with a good initial model. We collected 26 "short" observation sequences, where a short sequence is one where the robot goes from one topological node to the next. We trained a hierarchical model, a "hierarchical with reuse" model (in which all the low level submodels were trained separately) and a flat model as shown in Figure 7. For the "hierarchical with reuse" model we also collected a separate sequence of observations for every abstract state. The result in Figure 7 shows that the hierarchical with reuse model converges a lot faster, and the fit to the the data is very good right from the beginning due to the fact that the submodels were already trained. We say a model converges when the mean

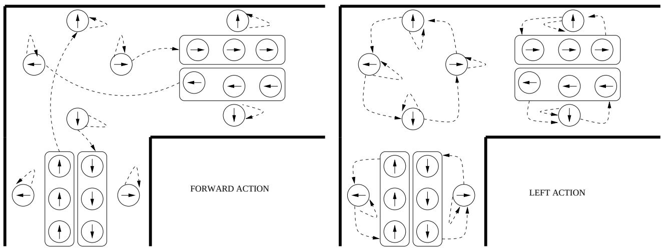

Fig. 4. These figures show how we represent a corridor environment as a hierarchical POMDP model. The circles represent product states and the solid arrows inside the circle indicate their orientation. The rectangles indicate abstract states (level 2) and the circles inside the rectangles are product states at level 3. The dashed arrows on the left show the transition matrix for the forward action at level 2 for a "good initial model". In the ergodic initial model (not shown), transitions are possible between all states under a given parent (for both abstract and product states). Before training we set all transitions to 0.5. The figure on the right shows the initial transition matrix for the "turn left" action. The "turn right" action is similar, except that the transitions are reversed.

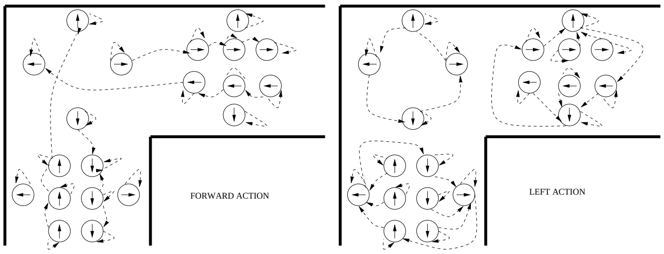

Fig. 5. These figures show the equivalent flat POMDP corridor model that can be automatically derived from the hierarchical POMDP model. The figure on the left show the transitions for the forward action and the one on the right shows the left action (both for good initial models).

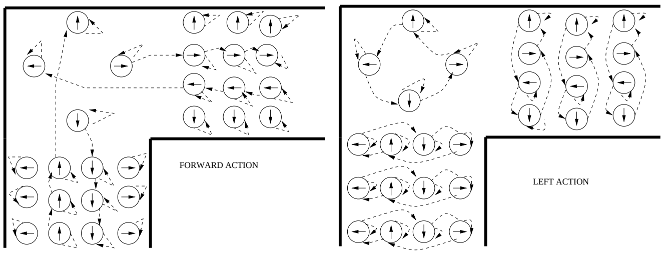

Fig. 6. These figures show how we represent a corridor environment as a flat POMDP model. The figure on the left shows the forward action and the figure on the right the turn left action (for good initial models).

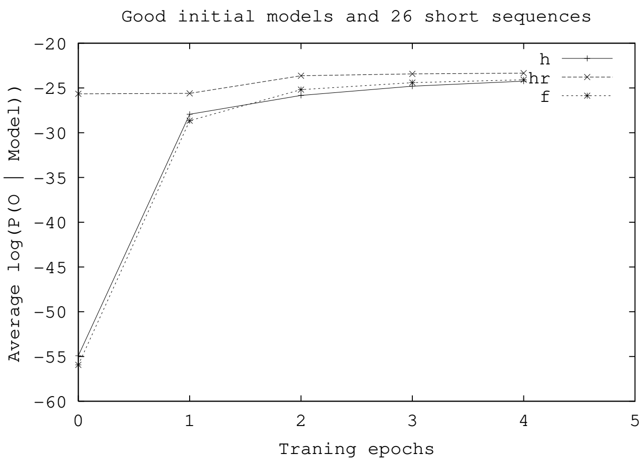

Fig. 7. The graph shows the training convergence between 3 different models. Model convergence is measured by mean square error across successive log likelihood values. The horizontal axis represents the number of training epochs and the vertical axis the average log likelihood fit of 26 different sequences of observations. *h* stands for a hierarchical model, *hr* stands for a hierarchical model whose initial model was created by first training separately the different ergodic models at level 3, and *f* stands for a flat model. Good initial models were provided for the training cases.

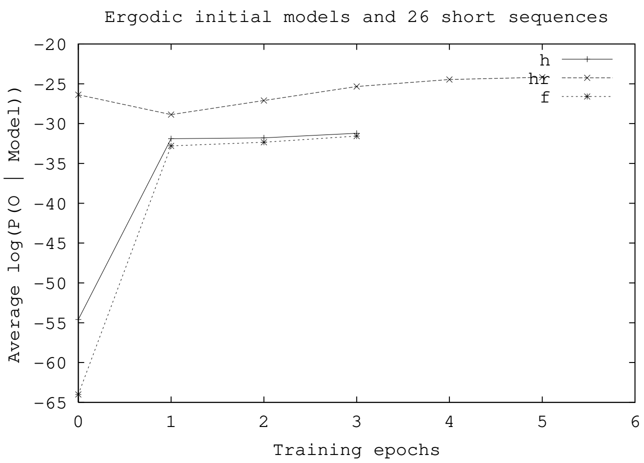

Fig. 8. The graph shows the training convergence between 3 different models. The X axis represents the number of training steps and the Y axis the average log likelihood fit of 26 different sequences of observations. *h* stands for a hierarchical model, where the starting model was ergodic, *hr* stands for a hierarchical model whose initial model was created by first training separately the different ergodic models at level 3, and *f* stands for a flat model. Here the starting model was also ergodic.

square error across successive log likelihood values for all training traces remains stable within some threshold.

In the second experiment, we compared hierarchical and flat POMDPs, beginning with weak uninformed ergodic models. The training convergence is shown in Figure 8. In this experiment, we see that the hierarchical with reuse model fits the data significantly better than the hierarchical and the flat models.

In a third experiment we trained the same models as those used in experiment 2 using one long sequence of 150 observations as shown in Figure 9. We observe that the hierarchical with reuse model and its equivalent flat model fit the data better than all the other models. The flat POMDP provides a poorer fit to the data than the other models. Table I shows both the size (number of states) of the various models and the time (number of seconds) it takes for a training epoch in each experiment. The hierarchical models and their equivalent flat models train faster than the flat POMDP model, partly due to the smaller number of states in these models as compared to the flat model.

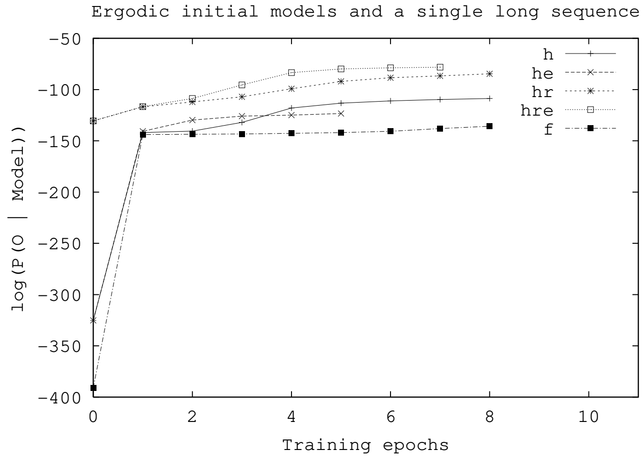

Fig. 9. The graph shows the training convergence between 5 different models for one long observation sequence. *h* stands for a hierarchical model, *he* stands for a flat model equivalent to the hierarchical, *hr* stands for a hierarchical with reuse model whose initial model was created by first training separately the different ergodic modules at level 3, *hre* is the equivalent flat model of the hierarchical with reuse model, and *f* stands for a flat model. For all cases, the starting models were ergodic.

	Figure 7, 8	Figure 9	Figure 11
Model	Time (Size)	Time (Size)	Time (Size)
h, hr	328 (346)	2200 (346)	300 (256)
he, hre	443 (286)	677 (286)	490 (232)
f	1416 (458)	21550 (458)	560 (410)

TABLE I

This table shows size of each model (number of states), and the time it takes (number of seconds) to train per epoch.

V. Real Robot Experiments

We also did experiments in an actual indoor environment as shown in Figure 10, where we collected 1 long sequence of a 100 observations using the mobile robot shown in the figure. The training convergence is shown in Figure 11, where the hierarchical with reuse model fits the data better than the rest of the models. The flat POMDP model once again provides the worst fit to the data and took the longest time to train, due to its size.

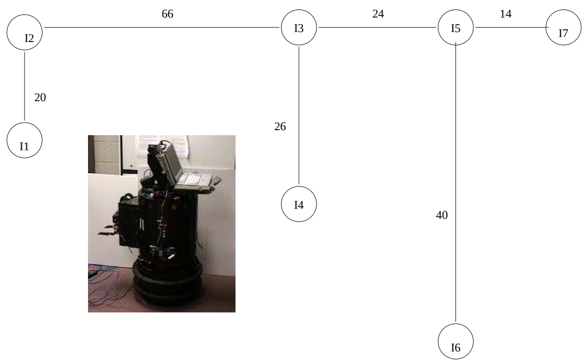

Fig. 10. The figure shows the topological map of a real indoor environment that is learned by PAVLOV (a Nomad 200 platform).

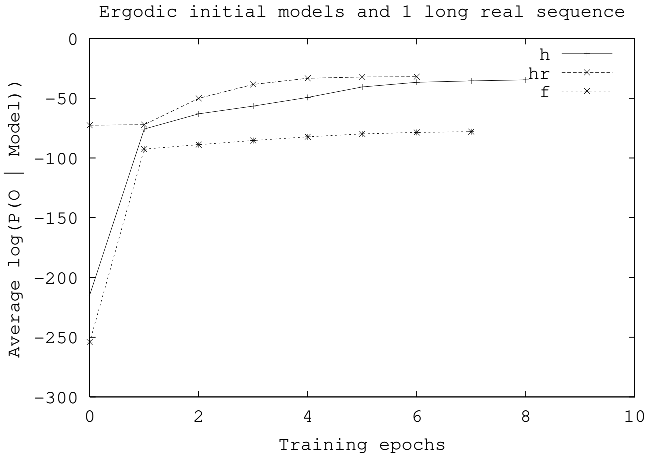

Fig. 11. The graph compares two hierarchical (*h* and *hr*)and one flat (*f*) model in terms of the goodness of fit and convergence (training epochs) for a real corridor environment.

VI. Conclusions and Future Work

In this paper, we describe a framework for learning hierarchical models of partially observable indoor corridor environments. We presented a modified Baum-Welch algorithm for learning hierarchical POMDP models. Using the new algorithm we compared flat and hierarchical POMDP models for both simulated and real world robot navigation environments. Our experimental results show that the hierarchical POMDP model is smaller in size, and correspondingly faster to train, even starting from a weak ergodic initial model. Furthermore, hierarchical models provide a natural way of reusing previously learned submodels. The fit to the data of the hierarchical models was also better than the flat model. We are now investigating alternate ways of defining abstract actions using the hierarchical nature of our POMDP model, and how abstract actions can speed up planning.

References

[1] Sven Koenig and Reid Simmons, "A Robot Navigation Architecture Based on Partially Observable Markov Decision Process Models," in *Artificial Intelligence Based Mobile Robotics:Case Studies of Successful Robot Systems.* MIT press, 1998.

[2] I. Nourbakhsh, R. Powers, and S. Birchfield, "Dervish: An office-navigation robot," in *AI Magazine 16(2):53-60.* 1995.

[3] Lawrence Rabiner, "Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," in *Proceedings of the IEEE*, February 1989, vol. 77.

[4] Hagit Shatkay and Leslie Kaebling, "Learning Topological Maps with Weak Local Odeometric Information," in *IJCAI97*, 1997.

[5] Shai Fine, Yoram Singer, and Naftali Tishby, "The Hierarchical Hidden Markov Model: Analysis and Applications," *Machine Learning*, vol. 32, no. 1, July 1998.

[6] Leslie Pack Kaebling, Michael L. Litman, and Anthony R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial Intelligence*, vol. 101, 1998.

[7] 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 (IROS)*, 1996.

[8] Sridhar Mahadevan, Georgios Theocharous, and Nikfar Khaleeli, "Fast concept learning for mobile robots," in *Autonomous Robots Journal, vol. 5, pp. 239-251.* 1998.