Variable Resolution Decomposition For Robotic Navigation Under a POMDP Framework

Robert Kaplow
School of Computer Science
McGill University
Montreal, Quebec, Canada
Email: rkaplo@cs.mcgill.ca

Amin Atrash
School of Computer Science
McGill University
Montreal, Quebec, Canada
Email: aatras@cs.mcgill.ca

Joelle Pineau
School of Computer Science
McGill University
Montreal, Quebec, Canada
Email: jpineau@cs.mcgill.ca

***Abstract*— Partially Observable Markov Decision Processes (POMDPs) offer a powerful mathematical framework for making optimal action choices in noisy and/or uncertain environments, in particular, allowing us to merge localization and decision-making for mobile robots. While advancements in POMDP techniques have allowed the use of much larger models, POMDPs for robot navigation are still limited by large state space requirements for even small maps. In this work, we propose a method to automatically generate a POMDP representation of an environment. By using variable resolution decomposition techniques, we can take advantage of characteristics of the environment to minimize the number of states required, while maintaining the level of detail required to find a robust and efficient policy. This is accomplished by automatically adjusting the level of detail required for planning at a given region, with few states representing large open areas, and many smaller states near objects. We validate this algorithm in POMDP simulations, a robot simulator as well as an autonomous robot.**

I. INTRODUCTION

Autonomous mobile robots are increasingly ubiquitous in society. Two of the major challenges in robotics are localization and navigation. Localization is the process of estimating the robot pose from sensor readings. While localization algorithms are becoming increasingly sophisticated, the task can still be challenging due to noise from sensors, actuators and the environment. The robot navigation task is to determine a path from one robot pose to another. Navigation techniques often assume the pose of the robot is known and do not take the localization uncertainty into account. Such systems use heuristics to ignore the uncertainty, such as simply assuming the robot is located at the most likely pose [26], or using voting techniques [24]. Thus, many robot navigation algorithms will disregard potentially useful localization information. By taking such localization uncertainty into account, a robot might take more useful long-term actions. For example, in cases of high uncertainty, the robot may choose to behave more pessimistically or possibly choose to take information gathering actions. To achieve this, one popular model is the Partially Observable Markov Decision Process (POMDP).

POMDPs are probabilistic models for decision-making in stochastic domains. In recent years, a number of more efficient approximate algorithms have been developed to solve POMDP problems [13], [16], [19], [22]. There has been previous work on applying POMDPs for robot localization and navigation [18], however, this work did not leverage these recent POMDP solution methods. For these tasks, the model is created by discretizing an environment map. Two popular methods used for map discretization are topological maps and metric maps [4]. For a POMDP model, the decision to use either a topological or a metric map is crucial, since the map decomposition will strongly affect the quality of the solution and the required planning time. Topological maps are high level abstractions of an environment, which allow navigation with a compact model. Metric maps discretize the environment spatially, using a set pattern such as a fixed grid.

The primary contribution of this paper is the application of variable resolution decomposition techniques to automatically produce a POMDP which can be solved to achieve a robust solution to the robot navigation task. By using a variable resolution decomposition, the spatial discretization of a given map can be generated automatically, and this will work with any sensor-built map. The variable resolution algorithm presented here has the advantages of both topological maps and metric maps. It is able to take advantage of the regular structure of indoor environments. This is done by identifying open spaces and abstracting them into a small number of states which represent large areas such as rooms and corridors. Conversely, areas near objects or walls get represented with a higher number of states. These are typically areas which require more detailed plans. Overall, this results in a smaller model with low resolution in large empty spaces and high resolution in dense spaces. The resulting POMDP model is superior for robot navigation. The high resolution discretization gives the robot higher precision near objects, giving the robot the ability to navigate effectively. Additionally, the reduced state space size will cut down the planning time dramatically, requiring the robot to take less time to construct plans, or improved plans for a fixed amount of planning time.

II. RELATED WORK

Having robust and accurate robot navigation algorithms is a crucial element of a fully autonomous robotic system, and has been a long term goal for robotics [8], [9]. One algorithm of note is the D* [23] algorithm, which focuses on creating optimal plans when the environment is only partially known.

Related algorithms have become more refined to work in more difficult environments [28]. However, in a real robotic system, applying a classical navigation technique in isolation is not enough, due to uncertainty in robot pose and noisy motion dynamics. For robust navigation, localization must be applied.

There are several classes of robot localization algorithms in general use. One popular family of localization algorithms uses Kalman filters [5], [6], which estimate the posterior distribution of robot pose using Gaussians. Another popular class of algorithms are particle filters [3], [27]. Particle filters, or Monte Carlo Localization, represent the belief about the robot's pose through a set of weighted samples, or particles. The particles are drawn from the posterior distribution of the robot's pose. Some advantages of particle filters are that the shape of the posterior distribution has no restriction, they are flexible in handling a variety of sensor characteristics, noise and movement dynamics, and they are easy to implement.

POMDPs are an ideal model for representing environments for robot navigation due to their ability to represent the noisy motion dynamics and noisy sensors. POMDP navigation was implemented on the Xavier robot [7], [17]. While this work was able to successfully use the POMDP model to localize the robot, heuristics were used to handle the navigation, due to the size of the model. In this case, topological maps were used, where the states represented rooms and sections of corridors. Roy and Thrun [14] did not use a POMDP directly, but approximated it by using an augmented state space which explicitly represents positional uncertainty as an extra dimension, and used this model to reduce uncertainty in a navigation task. Theocharous et al. [25] used hierarchical POMDPs for navigation in topological maps. In this work, POMDPs are used to learn the environment dynamics. For action selection, heuristics such as most likely state are used. Tomatis et al. [11] used a hybrid hierarchical approach, with a POMDP modelling the high level topological map, with a lower level composed of local metric maps. In [21], Spaan and Vlassis use a point-based POMDP solver for robot planning in an environment with a large set of pre-specified state locations. This work leveraged the newest POMDP approximation techniques, however, it assumed a known state space which might not scale well to large environments.

Variable resolution decomposition techniques for map discretization have been long established. The Quadtree algorithm [15] is a common technique for spatial representation in robotics, and have been used in navigation algorithms [29]. A binary space partitioning (BSP) tree [4] is a hierarchical structure that can be used for spatial representation. A cell in a BSP is either a leaf, or is recursively subdivided via a line parallel to the edges of the environment. Another approach is the exact method [4], which splits the space into non-overlapping regions via lines. Moore and Atkeson used an idea similar to Quadtree for a reinforcement learning task [1]. Zhou and Hansen [30] also used a variable resolution state space to reduce the size of a POMDP. However, the decomposition of this work is based on the value function itself rather than characteristics of the environment.

III. Background

A. Overview of POMDPs

The POMDP framework is a generalized model for planning under uncertainty [20], and is the partially observable analogue of the Markov Decision Process (MDP) [2]. A POMDP can be represented as a tuple $(S, A, O, T, \Omega, R, \gamma, b_0)$, where S is a (finite) set of discrete states, A is a (finite) set of actions, and O is a (finite) set of observations, which provide incomplete or noisy state information. The POMDP is further parameterized by:

$$\begin{aligned} T(s,a,s') &= P(s_{t+1} = s' | s_t = s, a_t = a), \\ \Omega(o,s',a) &= P(o_{t+1} = o | s_{t+1} = s', a_t = a), \end{aligned}$$

where $T(s,a,s')$ is the transition model, which describes the probability of transitioning to s' when the agent is in state s and undergoes action a, and $\Omega(o,s',a)$ is the observation model which describes the probability of receiving the observation o after taking action a and ending in state s'.

The function $R(s,a) \in \mathbb{R}$ is the reward for taking action a in state s, $\gamma \in [0,1]$ is the discount factor, and b_0 denotes a distribution over start states.

Under the POMDP framework, the agent has no direct knowledge of the current state. Instead, it must rely on observations for decision making. For the agent to make correct decisions, it summarizes its history into a belief state b, where b is a probability distribution over S which represents the agent's spatial belief:

$$b_t = P(s_t | b_0, a_0, o_1, ..., a_{t-1}, o_t). \tag{1}$$

The belief state is updated after every transition by Bayes' rule:

$$b_t(s') = \eta \Omega(o, s', a) \sum_{s \in S} T(s, a, s') b_{t-1}(s), \tag{2}$$

where η is a normalizing factor.

The goal of a POMDP solver is to determine a policy, π, which is a mapping from belief space to action space.

$$\pi : b \rightarrow A. \tag{3}$$

The policy attempts to maximize the value function, defined as the sum of discounted future rewards.

$$V(b) = E_\pi \left[\sum_{t=0}^{\infty} \gamma^t R_t \right], \tag{4}$$

where R_t is the reward received at timestep t, $V(b)$ is the value of the executing the policy starting at belief state, b, E_π is the expectation under policy π and $\gamma \in [0,1]$ is the discount factor.

In this work, we will take advantage of some of the recent advancements in POMDP solving, notably the work in the point-based approximations [13], [16], [19], [22]. These methods can solve POMDPs an order of magnitude faster than competing techniques, and are used in this work due to the large POMDPs required to represent real-world navigation domains.

B. Map Representations

For a robot to be able to efficiently navigate to a specific goal location in an environment, it is required for the robot to keep an internal representation of the environment, usually in the form of a map. To create a POMDP model for navigation tasks, a mapping from the environment map to the state space is required. Two popular choices of state space selection for navigating with POMDPs are metric maps and topological maps.

Metric maps are map representations which decompose the map spatially such as in a fixed grid. In each subcell of the map representation, it will measure if there is an obstacle or space, creating an occupancy grid. An important advantage of metric maps is their generality. They make no assumptions on what kind of environment the robot is working with. Therefore the transformation from the map to representation can be completely automatic. A further advantage of a fixed grid metric map is that they can be made to be very precise, since they require working with a fine resolution for robust navigation. However, a disadvantage of fixed grid metric maps is the steep model size requirements. This disadvantage is amplified when we use the model for POMDP planning, since the modern POMDP planning algorithms are polynomial in $|S|$.

Another popular map representation employed frequently for robot navigation are topological maps. A topological map explicitly represents the environment's connectivity information, e.g. in the form of a graph. A typical topological description of an indoor environment represents large abstract areas, such as rooms, as single states. Corridors are typically represented as a few states, which serve to connect the rooms in the topological graph. An advantage of using a topological map is that they provide much smaller models than metric maps. Additionally, they provide a human readable decomposition of the environment. However, using a topological representation can result in very coarse maps, which may not be suitable for very precise planning.

The Quadtree [15] algorithm creates a metric map discretization of an environment. Quadtree makes up for the shortcomings of the fixed grid representation by giving different resolutions for different areas in the map. The algorithm recursively subdivides a region in four equally sized quadrants until some stopping condition is reached. In the case of robotic navigation, usually the stopping condition is when a cell is fully occupied by an obstacle, or fully empty space. This method can be used for environments with varying levels of detail, for example, an environment with many large empty rooms but with some very narrow corridors.

IV. Variable Resolution Decomposition State Space

Using POMDPs for robot navigation has been historically difficult due to the high cost of finding a policy for a model with a large state space [14]. The most common approach is to use a fixed grid metric map representation, where each grid cell is a separate state in the POMDP model. This results in a very effective policy, since the belief state is precise due to each state being physically small. However, the resulting POMDP has a large state space. When $|S|$ is large, the POMDP is much more difficult to solve since the point-based POMDP solution techniques require $O(|S|^2)$ time. Therefore, we would like to construct a POMDP which will result in a usable policy while minimizing the size of the state space. To achieve this, we propose our variable resolution decomposition algorithm (Algorithm 1).

Algorithm 1 Variable Resolution

```
if ∃ space in section_of_map and
∃ obstacle in section_of_map and
section_of_map is larger than minimum_size then
    Split section_of_map along longer half into section_a
    and section_b
    Variable Resolution(section_a)
    Variable Resolution(section_b)
else
    stop
end if
```

Algorithm 1 can be seen as a modified version of the Quadtree algorithm. The input to this algorithm is the occupancy grid of the environment map. At each step of the recursion, it checks if the current section has both space and obstacle by checking each cell of the section in the occupancy grid. If the section has both, then it splits into halves by cutting along the longer edge. Otherwise, the section becomes a state in the resulting state space. Termination can also occur if the section is smaller than some minimum threshold size. If the minimum size threshold is reached for a section, then the resulting state can be considered to be occupied. This parameter can be used to control the size of the state space based on the requirements of the robot. In comparison to Quadtree, we cut only in half to keep the resulting state space as small as possible. This algorithm has a running time of $O(N \log(N))$ where N is the number of cells of size minimum_size in the occupancy grid of the environment map.

The goal of this technique is to reduce the size of the state space while keeping the model effective for robot navigation. This latter goal can be accomplished by decomposing the map with the following two design aims: to group together areas of the map where precise navigation is not critical, and to discretize areas near objects where navigation needs to be more precise with smaller resolution cells. These aims both relate to the key idea of grouping large areas of space and large areas of obstacle together. The outlined algorithm accomplishes this by recursively splitting up the map, and stopping when the current state is atomic (all space or object). Using the output of this algorithm as the state space for the POMDP model can greatly reduce the size of the model. An example of this algorithm's output can be seen in Figure 1.

From this figure, we can see that the algorithm assigns larger cells in larger spaces, where precision in navigation is

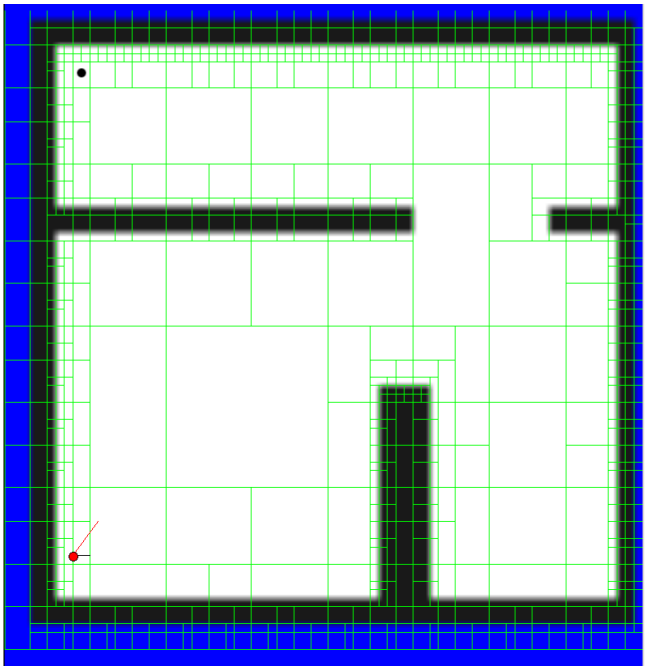

Fig. 1. A simple navigation task. We see the robot in the start location in the bottom left, and the goal in the top left. We can also see the variable cell decomposition output shown with the lines.

not as important, and assigns smaller cells in the corridors and near objects, as intended. A critical assumption of this technique is that all points within a single state have the same optimal policy. While degenerate cases of single states which require multiple actions can occur, these typically have a weak effect on overall performance due to the ability of the POMDP to recover from such mistakes. Since the algorithm decomposes the map to axis-aligned rectangles, the algorithm does best in rectilinear environments. However, we will see the algorithm is still able to drastically reduce the size of the state space in a variety of environments.

V. Description of the POMDP Model

While the state space of the POMDP is automatically computed by the variable resolution decomposition algorithm, the remaining parameters of the POMDP must now be defined.

A. Actions & Transition Model

For the POMDP, a set of high level actions must be selected. For example, in robot navigation tasks this can be defined as displacements in the four cardinal directions.

The actions and transitions are very robot dependent. While these could be constructed through domain knowledge, this is sometimes infeasible. An alternate method is the use of sampling techniques to approximate these parameters. We now describe the sampling method used in this paper. We build the next state distribution for executing action a in state s. First we initialize counts $\forall_s C(s) = 0$. We randomly draw a starting point p from within state s. We then apply the action a translation on p, and then add a small amount of Gaussian noise proportional to the length of a to the destination point p'. This noise is added to make a simple approximation of the noisy motion dynamics. Next, the straight line path between p and p' is verified to check that it does not pass through any obstacle state. This is required so the agent does not think an action can teleport the robot over an obstacle. If the straight line path is possible, then we find the state s' such that $p' \in s'$, and increment $C(s')$. This is done for many sampled points p, and then $C(s)$ is normalized to become $T(s, a, s')$. A primary advantage of building the transition model through sampling is that it makes no assumptions on either the environment or the type of spatial decomposition used.

B. Observations & Observation Model

As with the actions and transitions, the observation model is very robot dependent. While any reasonable observation model can be used, we present the model used in the experiments below. We choose an observation space which is an abstraction of information received from proximity sensors. We parameterize Ω by the resulting state s' only, and not the action a which resulted in the agent ending in state s', i.e. $\forall_{a \in A} \Omega(o, s', a) = \Omega(o, s')$. The observation is based on whether the adjacent states of s' are obstacle states or open space states. An edge of the state has a probability of being considered "filled" based on the proportion of that wall that is adjacent to obstacle states. This probability is perturbed with noise. The resulting model has sixteen observations, based on all possible combinations of the four walls being "filled" or "open", where the probability of a single observation is the product of the probabilities of the four walls being in their associated configuration. An advantage of this observation model is that it has only a small discrete number of possible observations, but it still provides observational information which will help the robot localize. However, a disadvantage of this model is that it uses only a very local measurement, since only the immediate vicinity affects the observation. It should be noted that this model is only used for policy building, and is not necessarily used for the navigation. On a real robot platform, the sensors are able to produce much more detailed observations.

C. Rewards

As per typical models for navigation tasks, the robot receives a fixed positive reward for reaching the designated goal states, and no reward otherwise. This leads to a policy which will get the agent to the goal as quickly as possible, presuming we are using discounting.

VI. Experiments

To validate our technique, a series of experiments were conducted on a variety of domains. The purpose of these experiments is to test the resulting quality of the produced policy based off POMDPs generated using the variable resolution techniques. Validation occurred in both simulation as well as on a robot platform. The test machine for these experiments was a dual core Xeon at 2.66Ghz with 4G of RAM.

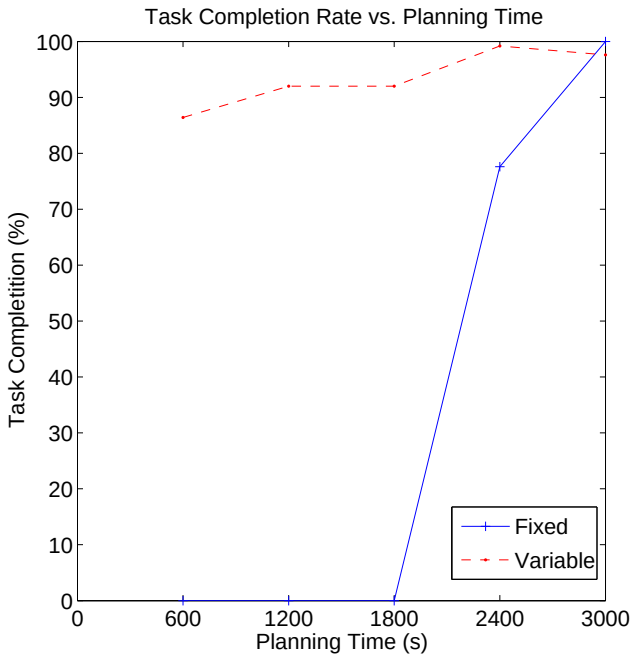

Fig. 2. Measuring the required planning time to complete the simple navigation task.

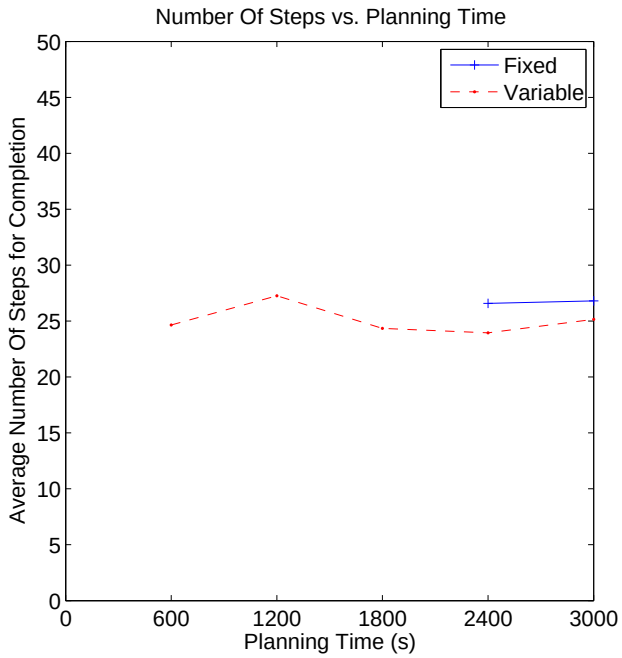

Fig. 3. Measuring the average number of steps to the goal in the simple navigation task.

A. POMDP simulation

The goal of using the variable resolution state space decomposition algorithm is to reduce the size of the state space, and thereby reduce the required time for producing a policy for a domain. A series of experiments were conducted to evaluate the performance of the policy produced by the variable resolution POMDP. At set time intervals, the policy was evaluated by POMDP simulation. For comparison, the same experiment was conducted on a fixed grid POMDP.

Figure 1 shows the original environment as well as the resulting decomposition. The fixed grid decomposition is not shown, but the size of the fixed grid cells is the size of the smallest cells in Figure 1. The number of states in the fixed and variable resolution POMDPs can be seen in Fig 4.

The dimensions of the fixed grid cell as well as the minimum_size parameter of Algorithm 1 is 10cm by 10cm. The actions for this domain are specified as the four cardinal directions, with two possible distances of 20cm and 80cm for each, giving us eight total actions. The transition noise is a Gaussian $N(0, .4|distance| + .1)$. The discount factor γ was set to .95.

A point-based approximate POMDP solver was used to solve both the fixed and variable resolution POMDPs.

For these results, we ran traces through the POMDP state space. Running the simulation for the fixed grid was straightforward, however, it should be noted that running the simulation for the variable resolution grid cells was a bit more complicated. In the variable resolution POMDP the state space grid cells are larger, so simply keeping track of the agent's actual state is not reliable enough. Instead, we consider traces through the fixed grid cells' state space, and we maintain our belief over the fixed grid states. Maintaining the actual state as a fixed grid cell provides a much more realistic experiment. Then at each step, we convert our fixed grid belief state into a variable resolution belief state. This is done by accumulating the belief of all fixed cells within a variable resolution cell as the belief for that variable resolution cell. Then the solved (variable resolution) policy, applied on the variable resolution belief state, is used to get the next action.

Because the POMDP solver has a randomized component, multiple policies were generated for evaluation. For these experiments, five policies were generated with 25 trials per policy. If an execution took more than 100 steps to complete, that trial was deemed a failure. The results from the simulations can be seen in Figures 2 and 3.

Figure 2 shows completion rate vs. planning time, where completion rate is the proportion of successful trials. We see that both methods are able to achieve the same success rate in the long term, since in this domain both methods are successful. Hence, even though the variable resolution algorithm loses precision with larger states this doesn't seem to hurt performance. We also see that it takes much less time to have a reasonably good policy in the variable resolution case. At roughly 600 seconds, the variable resolution policy has a high rate of success in the task, while the fixed policy agent is not able to complete the task at all until 2400 seconds.

Figure 3 shows the average number of steps vs planning time. This measure is the average number of steps taken in only the successful trials. We see that the average number of steps to the goal does not vary as the planning time

increases. This means that even preliminary policies provide near-optimal plans. Note that the number of steps to goal for the fixed policy is only shown after 2400 seconds. This is because the fixed grid POMDP policy was not able to find the goal in any trial until 2400 seconds of planning time.

B. Environment Map Survey

In the previous section, we show that variable resolution decomposition is able to reduce the required state space to model the environment of a toy domain, but still provide a practical policy. In this section we investigate the state space reduction capabilities on a set of maps generated by robots using sensor data. The maps freiburg, longwood, nsh_level_3 and thickwean are available with the CARMEN software package [10]. The results are shown in Figure 4.

Map Name	#Fixed	#Variable	Reduction Factor
Simple	3990	316	12.627
McConnell	29638	2035	14.56
freiburg	19226	3429	5.607
longwood	82595	8743	9.447
nsh_level_3	57003	5690	10.018
thickwean	14048	1488	9.441

Fig. 4. Comparing the number of grid cell for both a fixed grid decomposition and a variable resolution cell decomposition. The reduction factor is $\frac{\#Fixed}{\#Variable}$

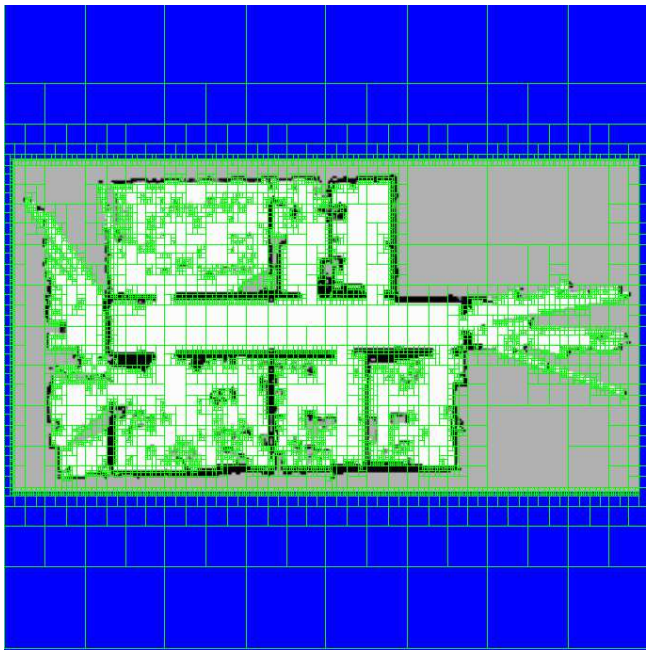

Fig. 5. The variable resolution decomposition algorithm applied to the freiburg map, released with the CARMEN.

These results show that the performance increase with this method is highly map dependant, however all maps show a sizable state space reduction. In a very wide open map like the simple environment, the reduction factor is large. On the other hand, in the freiburg (Figure 5) map, there are many small obstacles throughout the map. This causes the algorithm to produce many small grid cells next to the obstacles. This fits with our design, since these are locations where the robot requires extra precision (e.g. near walls, doors). These results are encouraging, since even with a state space reduction factor of five, the resulting POMDP should experience a 25-fold reduction in planning time. This reduction allows policies to be found for previously intractable POMDPs.

C. Test on SmartWheeler Robotic Wheelchair

For further verification of the method, we tested the algorithm on SmartWheeler [12] the robotic wheelchair, shown in Figure 6.

Fig. 6. The SmartWheeler Robotic Wheelchair

SmartWheeler is an electric wheelchair fitted with an onboard computer, motor actuators, SICK laser scanners and a touchscreen. The long-term goal of the SmartWheeler project is to increase the safety and autonomy of individuals with severe mobility impairments by designing a robotic wheelchair that can adapt to the user's needs and the constraints of the environment. We use CARMEN [10], the CMU Robot Navigation Toolkit, for basic robot control such as motor control and obstacle avoidance. Actions selected by the policy are passed to CARMEN, which translates them to motor commands that are sent to the robot. We also utilize the built-in obstacle avoidance of the CARMEN robot control system, since the POMDP planner does not account for dynamic obstacles. If the high level POMDP planner directs the robot towards objects not in the pre-computed plan, then CARMEN will navigate the robot around obstacles.

To take advantage of the SICK laser rangefinders, we use the built-in CARMEN particle filter for updating the belief. At each step, the new belief is computed by using the current set of particles as follows:

$$b(s) = \eta \sum_{p \in P_s} w(p) \tag{5}$$

where P_s is the set of particles that fall within state s, $w(p)$ is the weight of a particle from the CARMEN particle filter and η is a normalizing factor.

In these experiments, we do not use the robot's orientation as part of its state space. However, the method extends readily to incorporate the orientation, as does the planning phase. In our implementation, we use the CARMEN particle filter to estimate the orientation, and the noise from the orientation is folded into the transition probabilities.

Experiments were conducted in the McConnell building at McGill University, according to the map shown in Figure 7. Information concerning the state space can be seen in Figure 4. The parameters in this domain are as in the POMDP simulation experiments, with the size of the fixed grid cell and the minimum_size of the variable resolution algorithm being 10cm by 10cm, action distances of 20cm and 80cm and the transitional noise $N(0, .4|distance| + .1)$.

A policy was found by running the point-based POMDP solver on the model for two hours. It is worth noting that the fixed grid decomposition for this map was unable to be solved due to to memory constraints. Only the variable resolution model was usable for this task on this platform.

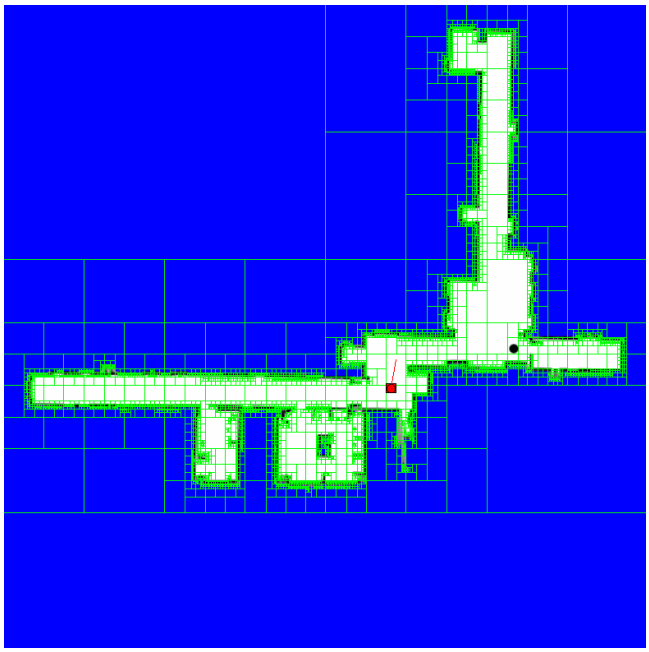

Fig. 7. The McConnell Navigation Task. The dark circle is the goal state, and we can see the square robot navigating through the environment.

Live robot experiments with the SmartWheeler robot show the algorithm to be successful. In all start locations we tested, the robot was able to correctly navigate to the goal location. This validates many of the assumptions used when building the POMDP model, for instance the abstracted observation model and the sampled transition model. Additionally, this shows the applicability of a mapping from the particles to the POMDP belief state.

Furthermore, the sensors used on the SmartWheeler, such as the laser rangefinger and the odometry are found on many robots. Therefore, we expect this approach to work on a variety of platforms.

VII. Conclusion

In this work, we present a variable resolution technique for robot navigation using POMDPs. This technique reduces the state space of the POMDP by automatically adjusting the size of the states in the grid based on features of the environment map. This allows us to gain the advantages of topological maps, such as reduced complexity/state space, while still maintaining the level of detail of fixed grid metric maps. This method allows us to apply POMDP navigation in domains which would otherwise have been infeasible. We validated this algorithm in a series of experiments including a POMDP-based simulation of a navigation environment, a realistic robot simulator and an autonomous robot. These experiments demonstrate the feasibility of the method in a variety of complex navigation tasks.

The partitioning heuristic used in this method considers only local information when considering the decomposition. A more sophisticated partitioning mechanism could produce improved policies with smaller state spaces. Another advantage of using a more global partitioning scheme is that it might better avoid the issue of states which require multiple actions as previously discussed.

The transition and observation models used for the POMDP model can be further refined. While the parameters in the sampling algorithms presented here were capable of providing good policies, better approximations to the motion dynamics and of the proximity sensors parameters will only improve the quality of the resulting policy. These models could be improved through learning, this is the subject on ongoing work.

While the results presented here are limited to wheeled mobile robots, we expect the technique to be applicable for robots with more degrees of freedom. These cases are particularly subject to an explosion in the size of the state space, when using fixed resolution, and thus stand to benefit substantially from the methods presented in this paper.

Acknowledgements

This research was supported by the Natural Sciences and Engineering Council of Canada and the Fonds Québécois de la Recherche sur la Nature et les Technologies.

References

[1] C. Atkeson A. Moore. The parti-game algorithm for variable resolution reinforcement learning in multidimensional state-spaces. *Machine Learning*, 21, 1995.

[2] R. Bellman. *Dynamic programming*. 1957.

[3] F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte Carlo localization for mobile robots. In *Artificial Intelligence*, 1999.

[4] G. Dudek and M. Jenkin. *Computational Principles of Mobile Robotics*. Cambridge University Press, April 2000.

[5] J. Gutmann and C. Schlegel. AMOS: Comparison of scan matching approaches for self-localization in indoor environments, 1996.

[6] R. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[7] S. Koenig and R. Simmons. Passive distance learning for robot navigation. In *International Conference on Machine Learning*, pages 266 – 274, 1996.

[8] J. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, 1991.

[9] S. LaValle. *Planning Algorithms*. Cambridge University Press, 2006.

[10] M. Montemerlo, N. Roy, and S. Thrun. Perspectives on standarization in mobile robot programming: The Carnegie Mellon Navigation (CARMEN) Toolkit. In *International Conference on Robotics and Systems*, pages 2436–2441, 2003.

[11] I. Nourbakhsh N. Tomatis and R. Siegwart. Simultaneous Localization and Map Building: A Global Topological Model with Local Metric Maps. In *Proceedings of the IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, 2001.

[12] J. Pineau and et. al. Smartwheeler: A robotic wheelchair test-bed for investigating new models of human-robot interaction. In *AAAI Spring Symposium on Multidisciplinary Collaboration for Socially Assistive Robotics*, pages 59–64, 2007.

[13] J. Pineau, G. Gordon, and S. Thrun. Point-based value iteration: An anytime algorithm for POMDPs. In *International Joint Conference on Artificial Intelligence*, pages 1025–1032, 2003.

[14] N. Roy and S. Thrun. Coastal navigation with mobile robots. In *In Proceedings of Conference on Neural Information Processing Systems (NIPS)*, volume 12, 1999.

[15] H. Samet. Region representation: quadtrees from boundary codes. *Communications of the ACM*, 23(3):163–170, 1980.

[16] G. Shani, R. Brafman, and S. Shimony. Forward search value iteration for POMDPs. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI)*, 2007.

[17] R. Simmons, J.L. Fernandez, R. Goodwin, S. Koenig, and J. O'Sullivan. Lessons learned from Xavier. *IEEE Robotics & Automation Magazine*, 7(2):33–39, June 2000.

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

[19] T. Smith and R. Simmons. Heuristic search value iteration for POMDPs. In *Proc. Int. Conf. on Uncertainty in Artificial Intelligence (UAI)*, 2004.

[20] E. Sondik. *The Optimal Control of Partially Observable Markov Decision Processes*. PhD thesis, Stanford University, 1971.

[21] M. Spaan and N. Vlassis. A point-based POMDP algorithm for robot planning. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2399–2404, 2004.

[22] M. Spaan and N. Vlassis. Perseus: Randomized point-based value iteration for POMDPs. In *Journal of Artificial Intelligence Research*, pages 195–220, 2005.

[23] A. Stentz. Optimal and efficient path planning for partially-known environments. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3310–3317, 1994.

[24] K. Tanaka, T. Hasegawa, Z. Hongbin, E. Kondo, and N. Okada. Mobile robot localization with an incomplete map in non-stationary environments. In *Proceedings of the 2003 IEEE International Conference on Robotics and Automation*, volume 2, Sept. 2003.

[25] Georgios Theocharous, Khashayar Rohanimanesh, and Sridhar Mahadevan. Learning hierarchical partially observable markov decision process models for robot navigation. In *Proceedings of the 2001 IEEE International Conference on Robotics & Automation (ICRA)*, pages 511–516, 2001.

[26] S. Thrun, W. Burgard, and D. Fox. A real-time algorithm for mobile robot mapping with applications to multi-robot and 3D mapping. In *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*, 2000.

[27] S Thrun, D. Fox, and W. Burgard F. Dellaert. Robust monte Carlo localization for mobile robots. *Artificial Intelligence*, 128(1-2):99–141, 2000.

[28] A. Yahja, S. Singh, and A. Stentz. *An Efficient On-line Path Planner for Outdoor Mobile Robots Operating in Vast Environments*, 33(1):129–143, August 2000.

[29] A. Zelinsky. A mobile robot exploration algorithm. *Robotics and Automation, IEEE Transactions on*, 8(6):707–717, Dec 1992.

[30] R. Zhou and A. Hansen. An improved grid-based approximation algorithm for POMDPs. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 707–716, 2001.