
Learning Action Models for Navigation in Noisy Environments

Natasha Balac
Daniel M. Gaines
Doug Fisher

NATASHA@VUSE.VANDERBIT.EDU
GAINESDM@VUSE.VANDERBIT.EDU
DFISHER@VUSE.VANDERBIT.EDU

Electrical Engineering and Computer Science Department, Station B, Box 1679, Nashville, TN 37235

Abstract

To be effective, a navigation planner must have knowledge not only of the effects an action will have, but also the effects that the environment will have on that action (e.g. the robot may travel more slowly over rough terrain). To address this issue, we have developed an approach called ERA which uses regression tree induction to learn action models that predict the effect terrain conditions will have on a robot's navigation actions. The action models support a high level planner that finds efficient navigation plans. We present the results of a study which evaluated the performance of ERA in environments with noisy effectors and sensors.

1. Introduction

An effective navigation planner for a mobile robot must be able to deal with large state spaces and take into account the uncertainty in the environment. One common approach to dealing with large state spaces is to use a high level planner that operates on a graph representing an abstract map of the area (Arkin, 1998). Each node in the graph corresponds to a region in the map and the edges reflect the topology of these regions. A navigation plan can be found with a search process (e.g. A*) to find the least cost path through the graph. The uncertainty in the environment is addressed by using reactive behaviors to navigate through each edge of the path. In summary, the high level planner uses deliberation to select appropriate way-points and reactive behaviors are responsible for getting the robot between these points.

While it is useful to use a high level planner to handle large state spaces, it is important that that planner use knowledge about the low level interaction of the robot. For example, the shortest path through the graph may not be the most efficient path if it requires the robot to traverse rocky terrain. We address this issue by having

the robot learn action models that predict the effects its navigation actions will have under different terrain conditions. For example, the robot will learn an action model for MoveForward that predicts how far the robot will move ahead in a single time step given the current terrain conditions. These predictive models are used to weight the edges of the high level planning graph. As a result, the least cost path through the graph represents the most efficient path given how fast the robot can expect to travel through the various terrain conditions.

We have developed a system called ERA that allows a robot to learn action models for its navigation actions. We call the approach ERA for its three main components: Exploration, Regression tree induction and Action models. With this approach, a robot explores the world to gather data about the effects of its actions. As it explores, it keeps track of the state of the world, the action that it attempted and the new state that resulted. We apply a regression tree induction algorithm to the data acquired for each action type to build a regression tree representing the model of that action. Regression trees offer the advantages of being able to predict a continuous variable and the ability to model noise in the data. In this approach, each regression tree predicts the expected outcome for an action based on environmental conditions.

The approach assumes that the robot can determine what terrain it is on when it takes an action. We assume the robot has a map of the area describing the terrain at each point and that it can use GPS to determine its position in the map. The United States has recently discontinued the Selective Availability (SA) feature of GPS giving civilian applications access to a more accurate signal. While this dramatically improves the positioning accuracy, it is still not perfect and there will be errors in the robot's prediction of its current position. This will lead to classification errors in the training data used by ERA. For example, the robot may think it is still on a smooth road when it takes a sample, when in fact it has gone off the road

and is in a field of low grass.

We present the results of a study in which we evaluated the effect of sensor noise on ERA. The results indicate that ERA can tolerate low sensor noise that is within the capabilities of current technology. However, higher noise levels impair ERA’s ability to learn accurate action models and support the high level planner.

2. Regression Tree Induction

Regression tree induction is a well-known approach for improving predictions along a continuous, dependent dimension (Breiman et al., 1984). A regression tree predicts a value along the dependent dimension for all environmental observations, in contrast to a decision tree, which enables a prediction along a categorical variable (i.e., class). As with decision trees, recursive partitioning is typically used to construct a regression tree to fit training data.

In the basic regression tree construction method of (Breiman et al., 1984), the tree is expanded at each prospective split point by splitting on the values of the independent variable that most reduces the mean-squared error (or some other measure of variance) along the dependent dimension. Leaves are labeled by the mean (or median) value among the training data that satisfy the conditions along the path to that leaf. A notable extension to this basic strategy allows *residual nodes* to be created, as well as traditional split nodes (Lubinsky, 1995). A residual node is the best-fitting 1st-order model relating an independent dimension with the dependent dimension.

Other approaches (Torgo, 1997; Quinlan, 1993) allow for alternative models other than regression models (e.g., kernal, nearest neighbor) to be used at regression tree leaves. These extensions may lead to advantageous results, but for now we use the basic strategy, coupled with a pruning strategy that cuts off path expansion when the *mean relative error* (MRE) falls below a user-specified threshold. The MRE is defined as the mean of $|M_a - M|/M_a$ over all M_a at a node, where each M_a is the value of the dependent dimension of an observation at a node, and M is the mean over all such observations at the node.

3. Overview of ERA

ERA allows a robot to learn predictive models of its navigation actions from experience. Figure 1 summarizes the approach. We call the system ERA because it consists of Exploration and Regression tree induction to produce Action models. During the exploration

phase the robot collects data using a simple wander program to navigate through the environment.

Currently, we are experimenting with ERA in simulation based on Georgia Tech’s MissionLab (MacKenzie et al., 1995). In its original form, MissionLab allows a robot to move about the environment and sense obstacles and other robots. We have extended this environment by creating terrain elements and adding noise to robot actions. In this paper we focus on the MoveForward action as it is used to support the high level planner.

Because we are using a simulation, the input to the system includes the map of the area and error models that determine how the robot’s actions will behave in the different terrain conditions. We ran some experiments with a simple Lego robot on different types of surfaces to get a feel for appropriate probability distributions to use to model noise in the MoveForward action. We found that normal distributions were adequate for approximating the error we observed. It should be noted that, although the simulator requires the error models, the robot itself does not observe these models directly. Instead, it samples the models as it explores the area. Of course, with a real robot, we will not have to create these error models. Instead, the robot will take samples from exploring the actual terrain.

With each exploratory step, the robot records the action that was taken, the current terrain conditions and its heading and position before and after the action. When exploration is finished, we divide the data into groups based on action type. We apply a regression tree induction algorithm to each data set to obtain a set of regression trees, one for each action type. Each tree models the action by specifying the expected outcome for the action given the different environmental conditions. The internal nodes in the tree contain environmental conditions that were found to influence the outcome of the action. The leaf nodes give the average outcome that can be expected if it is attempted under those conditions.

3.1 Example of Learning Action Models

Figure 1 shows the results of ERA on an example map. The example includes four different types of terrain each with a different roughness. The four terrain types correspond, in order of increasing roughness to: road (R=0), grass (R=1), dirt (R=2) and rocks (R=3). The error models used for each terrain condition are shown in the lower left corner of the figure.

After the exploration stage, the regression tree induction algorithm produces the tree in the far right of the

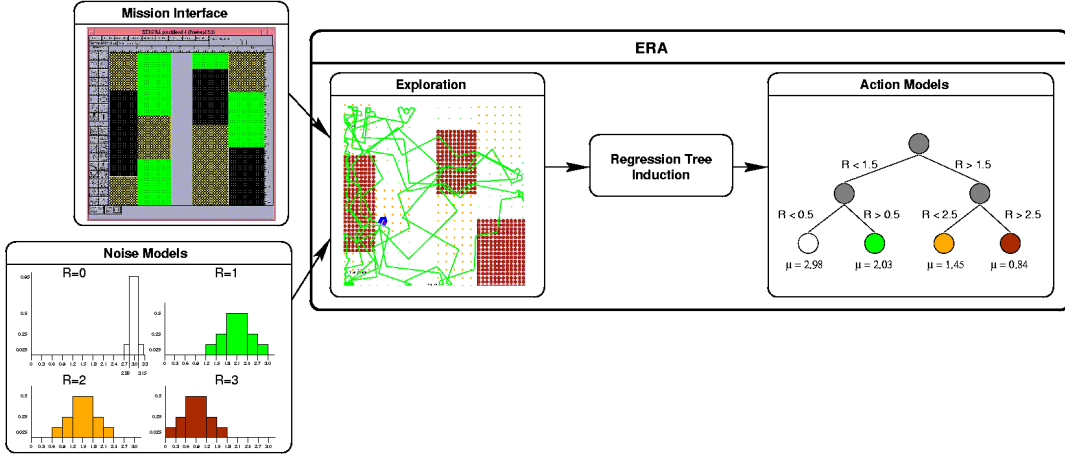


Figure 1. Overview of ERA.

figure. The tree has four leaf nodes, each corresponding to a different roughness level. Each node represents a prediction of how far the robot will travel over that type of terrain in a single time step. For example, the far left node predicts that the robot can travel 2.98 m over a road while the far right node indicates the robot can travel 0.84 m over rocky terrain.

3.2 Example of Supporting a High Level Planner

Figure 2 shows how the action models are used to support the high level planner we call Mission Planner. In the first step an abstract graph is generated from the terrain map. The graph is created by placing a node at the center of each terrain element. Nodes are then placed at the center of each edge on the terrain element’s border with an edge connecting to the node at the center. This is done for each terrain element and results in the graph shown in Figure 2. An edge is weighted to reflect how many time steps it is expected to take to travel between the corresponding nodes. The weight, w , is calculated as: $w = \frac{d}{\mu_r}$ where d is the Euclidean distance between the center of the two nodes and μ_r is the average distance traveled in one time step over terrain with roughness value r .

The user specifies a starting point and ending point for the mobile robot. We use uniform cost search to find a least cost path in the graph leading from the start point to the end point. The path found for this example is shown in the far right of Figure 2. Note that this is not the shortest path, which would have traveled over rocky terrain. Instead, the planner found that it is more efficient to move over to the road, follow

the road for a while and then move back to the goal.

These action models can be applied to other problems that have the same types of terrain elements. So, for example, a user can create a new map using different sizes of rock, dirt, grass and road terrain elements in different locations and the planner can use the pre-learned action models without re-running ERA. Thus, the robot can transfer the knowledge that it learns to related problems.

4. The Effects of Sensor Noise on ERA

We have described ERA assuming that the robot will always know what type of terrain it is on by using GPS to determine its current position. Of course, even with an accurate map, this is an unrealistic assumption given the error in GPS. Therefore, we wanted to find out how relaxing this assumption would affect ERA. In particular, we were interested in finding out how increasing sensor noise would impact the accuracy of the action models and the cost of the resulting plans.

We expect that as noise increases, the accuracy of the action models will decrease. Although we expect the plan cost will increase as the planner uses less accurate models, we are hoping that the cost will degrade slowly since the plans depend more on the relative values across terrain conditions than on precise values.

Currently, we only introduce noise in the robot’s ability to classify terrain and not its ability to determine how far it traveled when it generates a sample. Of course, this type of noise is likely to effect the robot’s ability to learn action models and we will study the effect in later work. We do, however, include variance

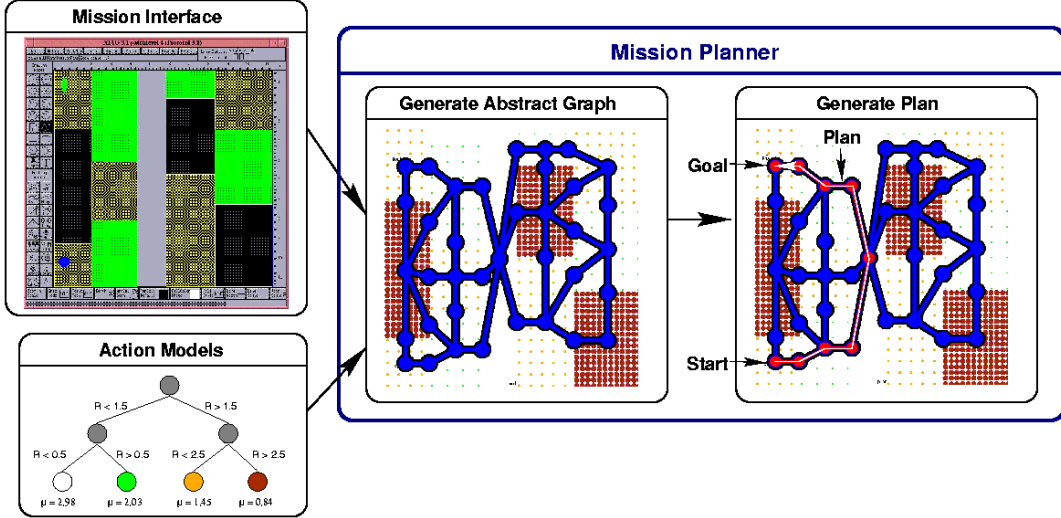


Figure 2. Overview of Mission Planner.

in the error models for the MoveAhead action and it may be the case that encoder noise would not alter these distributions significantly.

4.1 Methodology

Figure 3 shows the terrain map used in the studies. The error models for the MoveAhead action are given in Figure 1. We introduced GPS sensor noise by impairing the robot’s ability to detect its current position, and thus, the type of terrain it is on. We consider several levels of increasing noise. At level n the robot’s location is known within a resolution of n meters. When it records its current terrain for a training sample, we get the list of all terrain elements within n meters of its current position and randomly pick one of these elements to be the terrain that the robot thinks it is on. Thus, at level 1, if the robot is within 1 meter of the boundaries of two different terrain elements, there is a 50-50 chance it will think it is in either terrain. If it guesses wrong it will have an erroneous terrain classification in its training data. As we increase the noise level, there will be more invalid terrain classifications in the training data.

We ran ERA at noise levels from 0 to 10 meters (with 1 meter increments). For each run, we recorded the regression trees that were generated after every ten samples up to 1000 samples. We also kept track of how many classifications appeared in the training data as sensor noise increased. Figure 4 shows the probability that a given data point in the training set was invalid for each noise level. The graph shows that each level

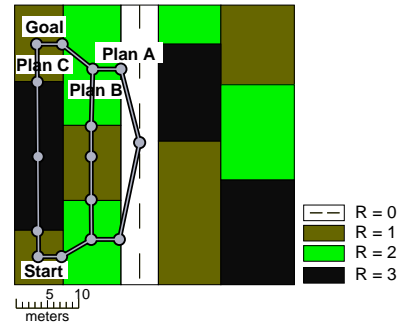


Figure 3. Map used for experiments.

increased the number of invalid classifications in the training data.

4.2 Effect on the Accuracy of Action Models

For each action model, we calculate the Mean Relative Error (MRE) for each unique terrain condition. MRE is given by: $MRE = \frac{|\mu_p - \mu_a|}{\mu_a}$, where μ_p is the predicted mean for this terrain condition and μ_a is the actual mean for that terrain as given by the error distributions in Figure 1. The closer the MRE is to 0, the better this action model is doing at predicting the MoveAhead action for this type of terrain. Since there are four types of terrain in the experiment, we calculate an MRE for each type and get the average MRE for this action model. Figure 5 shows the average Mean Relative Error (MRE) for the action models generated after 1000 steps for each noise level.

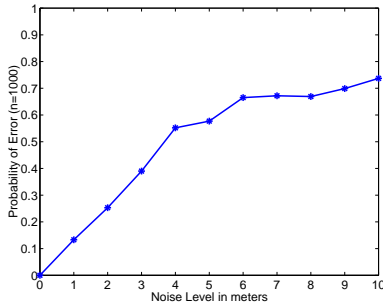


Figure 4. Probability of a given data point being incorrectly classified in the training data.

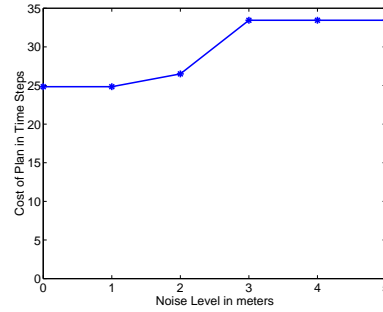


Figure 6. Cost of plan.

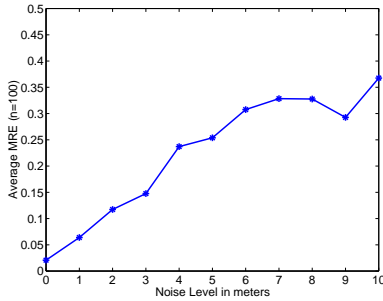


Figure 5. Average Mean Relative Error (MRE).

As expected, as noise increased, the accuracy of the action models decreased. We observed that, as noise increased, the predicted means for each terrain converged on the global mean for all error models (i.e. if we were to combine all of the error distributions and calculate the mean of this composite distribution). This makes sense because as sensor noise increases, the samples classified as a particular terrain condition will include more samples from other terrain conditions, in effect merging the distributions.

4.3 Effect on the Cost of the Resulting Plans

We used the action models from above to generate plans for moving from the center of the lower left region in Figure 3 to the center of the top right region. Figure 6 shows the cost of the resulting plan, in time steps, as the noise level is increased. Both the 0m and 1m error levels were able to find the plan with the least expected cost; Plan A from Figure 3. This plan is not the shortest, but it takes advantage of the fact that the robot can move more quickly over the road. At noise level 2m, the robot finds Plan B which is not as efficient but still avoids the roughest terrain. In the rest of the plans, the robot travels the shortest path, Plan C, which takes it through the roughest area.

Although increasing noise had a negative impact on the cost of the plan, the robot was still able to find the best plan at the 1m level and the second best plan at the 2m level. It is possible to get sub-meter accuracy with differential GPS and we can also incorporate other sensor modalities to help differentiate terrain.

It is important to note that the effect of noise on the training data is dependent on the size of the terrain elements. With very small terrain regions, 1 meter of noise would lead to a much larger number of invalid terrain classifications in the training data. Conversely, larger regions would lead to fewer classification errors. Thus, Figure 4 is a better indication of ERA’s tolerance to noise. It shows that at 1m of error, about 13% of the training data was improperly classified, yet the system was still able to find the best plan.

5. Related Work

There has been a variety of approaches to learning action models for planners. The focus of the work, as in this paper, is to address the problem of acquiring knowledge about the world for a planner.

Gil presents an approach to refining a planner’s incomplete knowledge base (Gil, 1994). Given a knowledge base consisting of STRIPS-style operators, her approach uses exploration and experimentation to fill in missing pre-conditions and effects. Wang’s OBSERVER system (Wang, 1995) is able to work from an empty knowledge base, using traces from experts’ problem solving to fill in the empty operator descriptions using an approach similar to the version method (Mitchell, 1982). Mahadevan showed how learning action models can enable transfer in reinforcement learning (Mahadevan, 1992). One of the limitations of these approaches is that they assume pre-conditions are conjunctive and they do not handle domain noise.

Benson’s TRAIL system extends this work by allow-

ing disjunctions in the pre-conditions and noise in an action's effects (Benson, 1995). TRAIL is able to learn the probability that an outcome of an action will occur but it only allows boolean outcomes. Furthermore, it does not allow the probability of the outcome be dependent on the pre-conditions. In contrast, ERA learns a distribution for the outcome of an action and learns how the pre-conditions will impact the outcome. desJardin's PAGODA system is also able to learn relationships between values of the pre-conditions and the probability that the action will succeed (desJardins, 1994). However, like TRAIL, it only learns whether or not the action will be successful.

Finally, variations on regression tree induction that allow for more expressive node models than binary splits are desirable. For example, if error grows linearly with roughness, then such extensions would more naturally and accurately capture the error distribution conditioned on roughness (Lubinsky, 1995; Torgo, 1997).

6. Conclusion

We have proposed a new approach, called ERA, that allows a robot to learn action models for predicting the effect of navigation actions under various terrain conditions. We have evaluated the performance of the approach under various levels of sensor noise. The results show that at lower levels of sensor noise, ERA can support a planner in finding low cost plans. As noise increases, the cost of the plan rises. However, the lower levels of noise are achievable with current GPS technology.

In future work, we plan to extend the regression tree induction algorithm to make multi-variate predictions. We will also be using ERA and MissionPlanner with a real robot.

7. Acknowledgments

This work was sponsored by the Defense Advanced Research Projects Agency grant DASG60-99-1-0005 issued by the U.S. Army Space and Missile Defense Command. We would like to thank the Mobile Robot Lab at Georgia Institute of Technology for creating and making available MissionLab.

References

Arkin, R. C. (1998). *Behavior-based robotics*. MIT Press.

Benson, S. (1995). Inductive learning of reactive action models. *12th International Conference on Machine*

Learning

Breiman, L., Friedman, J., Olshen, R., & Stone, C. (1984). *Classification and regression trees*. Belmont, CA: Wadsworth.

desJardins, M. (1994). Knowledge development methods for planning systems. *Fall Symposium on Learning and Planning: On to Real Applications* (pp. 34–40). New Orleans.

Gil, Y. (1994). Learning by experimentation: incremental refinement of incomplete planning domains. *Eleventh International Conference on Machine Learning* Rutgers, NJ.

Lubinsky, D. (1995). Tree structured interpretable regression. In D. Fisher and H.-J. Lenz (Eds.), *Learning from data: Artificial intelligence and statistics v*, 387–398. New York: Springer.

MacKenzie, D., Cameron, J., & Arkin, R. C. (1995). Specification and execution of multiagent missions. *Proceeding of the International Conference on Intelligent Robotics and Systems (IROS '95)* (pp. 51–58).

Mahadevan, S. (1992). Enhancing Transfer in Reinforcement Learning by Building Stochastic Models of Robot Actions. *Ninth International Conference on Machine Learning* (pp. 290–299). Aberdeen, Scotland.

Mitchell, T. M. (1982). Generalization as search. *Artificial Intelligence*, 18, 203–226.

Quinlan, J. R. (1993). Combining instance-based and model-based learning. *Proceedings of the Tenth International Conference on Machine Learning* (pp. 236–243). Amherst, MA: Morgan Kaufmann.

Torgo, L. (1997). Functional models for regression tree leaves. *Proceedings of the Fourteenth International Conference on Machine Learning* (pp. 385–393). Nashville, TN: Morgan Kaufmann.

Wang, X. (1995). Learning by observation and practice: An incremental approach for planning operator acquisition. *12th International Conference on Machine Learning*