SITUATIONAL REACTION AND PLANNING

John Yen and Nathan Pfluger
Department of Computer Science
Texas A&M University
College Station, TX 77843-3112

Abstract

One problem faced in designing an autonomous mobile robot system is that there are many parameters of the system to define and optimize. While these parameters can be obtained for any given situation, determining what the parameters should be in all situations is difficult. The usual solution is to give the system general parameters that work in all situations, but this does not help the robot to perform its best in a dynamic environment. Our approach is to develop a higher level situation analysis module that adjusts the parameters by analyzing the goals and history of sensor readings. By allowing the robot to change the system parameters based on its judgement of the situation, the robot will be able to better adapt to a wider set of possible situations. We use fuzzy logic in our implementation to reduce the number of basic situations the controller has to recognize. For example, a situation may be 60 percent open and 40 percent corridor, causing the optimal parameters to be somewhere between the optimal settings for the two extreme situations.

Introduction

The design and implementation of autonomous mobile robot planning and control systems will allow robots to handle tasks that are very dangerous or impossible for a human controlled system to handle correctly. Instances of these tasks are the autonomous rover system for the exploration of Mars, and the battlefield tank controller, where ECM may make it impossible to remote control tanks in enemy territory.

On problem faced in designing an autonomous mobile robot system is that there are many parameters of the system to define and optimize. Some of these parameters include maximum safe speed, how near obstacles can be without being threats, and how far the robot can safely project its destination. While these parameters can be obtained for any given situation, determining what the parameters should be in all situations is difficult. The usual solution is to give the system general parameters that work in all situations, but this does not help the robot to perform its best in a dynamic environment. The problem of finding parameters is further complicated if the robot is able to perform many missions, like exploring or transporting, which each require the robot to behave differently in identical environments.

Our approach is to develop a higher level situation analysis module that adjusts the parameters by analyzing the goals and history of sensor readings. By allowing the robot to change the system parameters based on its judgement of the situation, the robot will be able to better adapt to a wider set of possible situations. We are currently implementing the situation analysis module using fuzzy if-then rules. Use of fuzzy logic allows us to specify less base situation and to combine these situations into more types of situation easier than a standard logic based system.

The rest of this paper is organized as follows. First we will give a brief overview of fuzzy logic and fuzzy control. We will then give an overview of intelligent systems. We will then cover situation recognition and its potential benefits. We will then discuss our testbed and an implementation of situation recognition in it. We will then give some results from simulations of our mobile robot controller.

Background

In this section, We will first give a brief overview of fuzzy logic and the use of fuzzy logic in decision making and control. Afterward we will discuss the development of intelligent systems.

Fuzzy Logic

Motivated by the observation that many concepts in the real world do not have well defined sharp boundaries, Lotfi A. Zadeh developed fuzzy set theory that generalizes classical set theory to allow objects to take partial membership in vague concepts (i.e. fuzzy

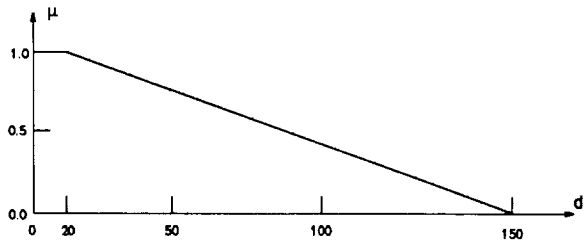

Figure 1: A Fuzzy Set Representing the Concept of Near

sets).[10] The degree an object belongs to a fuzzy set, which is a real number between 0 and 1, is called the *membership value* in the set. The meaning of a fuzzy set, is thus characterized by a *membership function* that maps elements of a universe of discourse to their corresponding membership values.

Figure 1 shows the membership function of the fuzzy set **NEAR** in the context of mobile robot navigation control. In this Figure, d represents the distance of the closest obstacle detected by a sensor, and μ represents the membership value in the fuzzy set **NEAR**. As the Figure depicts, an object that is 15 units away has a full membership in **NEAR**, while one that is 50 units away has a membership value of 0.8. Capturing vague concepts such as **NEAR** using fuzzy sets can improve the robustness of a navigation control system in the presence of sensor noise, because noise in the sensor data can only slightly change the membership degree in **NEAR** and therefore affects the final control command in a minor way.

Based on fuzzy set theory, fuzzy logic generalizes modus ponens in classical logic to allow a conclusion to be drawn from a fuzzy if-then rule when the rule's antecedent is partially satisfied. The antecedent of a fuzzy rule is usually a boolean combination of fuzzy propositions in the form of "x is A" where A is a fuzzy set. The strength of the conclusion is calculated based on the degree to which the antecedent is satisfied. A fuzzy logic controller uses a set of fuzzy if-then rules to capture the relationship (i.e. the control law) between the observed variables and the controlled variables. n each control cycle, all fuzzy rules are fired and combined to obtain a fuzzy conclusion for each control variable. Each fuzzy conclusion is then *defuzzified*, resulting in a final crisp control command. An overview of a fuzzy logic controller and its applications can be found in the work by C.C. Lee.[3, 4]

Intelligent Systems

We are presently delving into the use of fuzzy logic in the implementation of intelligent systems. A definition of an intelligent system from a paper by Albus[1] is that an intelligent system must at least have the ability to sense the environment, make decisions and to control actions. Higher levels of intelligence may require the

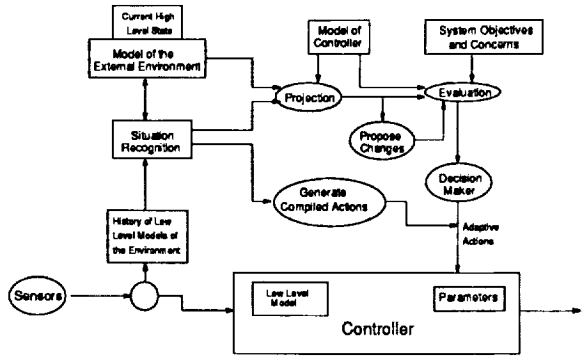

Figure 2: High Level Model of Situation Recognition and Adaptation

recognition of objects, the storage of knowledge for future uses, and the ability to reason how actions will affect the future.

All of these functions will be needed by a system that wishes to act in complex dynamic environments effectively. An example of such an application is an autonomous mobile robot system. The lower level abilities of sensing and control are used by the robot to quickly sense and avoid obstacles in the path. The higher level abilities of recognition and projection are needed to allow the robot to adjust itself to any environment.

We have used fuzzy logic to implement a behaviorial based system that is able to use sensors to control the robot to follow a given path while avoiding obstacles in the path, a discussion of which is given in the next section. We are currently working on using the ability to recognize elements of the environment and to reason about how those elements will effect the robot in order to compute the most effective parameters to use in any given situation.

Situation Recognition

In this section a general architecture for including situation recognition is given. We will then overview our method for situation recognition and reaction of the system to recognized changes in the situation.

General Architecture

The general architecture we have adopted for situation recognition and reaction is given in Figure 2. The main concept is to develop a system independent of the controller that has the ability to adapt the controller based on its perception of the situation, i.e. a metalevel controller. The architecture has one set of inputs, the sensors, and produces a set of adaptive actions for the controller.

The sensors are fed into a situation recognition architecture along with a high level model of the environment. This purpose of this module is to examine the sensor histories and the changes in the high level model

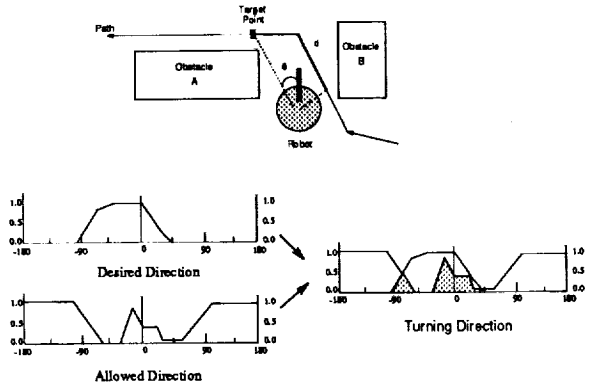

Figure 3: Desired and Allowed Directions

to determine if there has been a significant change in the environment to warrant adaptive action for the controller. How this module operates is given in the next subsection.

Once a new situation has been recognized, there are two possible methods of handling the change. The first is to directly generate precompiled actions for those changes. These actions are then fed to the controller for adapting to the current situation.

The other method of adaption is to use a form of projection to find the possible effects of the change in environment. If the effects are large enough, then changes need to be proposed and the new system is projected till a satisfactory projection is found.

The architecture given has not been fully implemented, and only represents the final system we want. At present the method of adaption is to use a set of precompiled actions to change the controller based on the perceived situation.

Testbed

We are currently implementing the situation recognition architecture on an autonomous mobile robot control system. In this section, we will give a brief overview of autonomous mobile robot systems and the fuzzy logic based behavioral architecture which we are currently using.

The basic problem of autonomous mobile robot path planning and control is to navigate safely to one or several target locations. The problem can be further complicated by other considerations such as deadlines for reaching those locations, safety considerations of paths, reactiveness to emergent situations and uncertainty about the environment. There are several approaches that accomplish this goal. We will concentrate on a behavioral implementation that uses fuzzy logic to merge sensor readings with path information to determine the final control command each cycle.[8, 9]

The approach we have taken uses fuzzy logic to describe the desired and allowed direction of travel, see Figure 3. The algorithm first determines the target point by projecting along the path given. It then uses fuzzy logic to broaden it into the desired direction. Next, the inputs from sonar sensors fixed around the body are fuzzified and combined to form the allowed direction. These two concepts are then combined to form a fuzzy control command that describes what the robot should do. After using Centroid of Largest Area (CLA) defuzzification,[7] a control command for the robot can be found. For a more detailed discussion, please see Yen and Pfluger.[8, 9]

The above method, as is the case with most sophisticated control systems, has a problem in specifying the control parameters to handle all situations. Some of the parameters whose optimality can be dependent on the environment include distance to target point, maximum speed for both going straight and turning, and *nearness* of objects for both forward and side sensors. When the robot was first programmed a set of values was taken such that robot would be able to work in any environment, i.e. very conservative values. The goal of this research is to improve the performance of the robot by changing the values of the control parameters in reaction to perceived changes in the environment.

Implementation of Situation Recognition

The first step is to determine what are the salient features of the situation that we want to recognize. At present we attempt to find three features:

1. *Openness:* This gives a general measure of the number of obstacles, both seen and expected.
2. *Path Information Strength:* Is the path along a road or a corridor? Should the path be followed religiously or just be taken as a possible path.
3. *Degree of Exploration:* Are we exploring the environment or traversing it? Or maybe a little of both.

Determining the amount of openness can be determined directly from the sensors. The method we are currently using is to use the allowed direction computation from the controller. This fuzzy set is determined by finding the nearness of obstacles and combining the results. By taking the fraction of the area that is allowed, we get a good measure of openness. This fraction can be further modified by the current values of nearness, i.e. if the nearness of obstacles is tight then the openness of the environment is less open.

The path information strength can be found both from the sensors and the path planning module. If the area is not open then the path strength should be stronger. If the goal of the robot is to explore, then the path strength should be less. The degree of certainty that the path is safe is passed from the planning module and influences this value. By using fuzzy rules to combine these and other concerns, a relative strength of the path can be found.

Finally, the degree of exploration can be determined from two sources. First, if the overall goal of the robot

is specified then the degree of exploration is that. This can be modified by the number of undetermined obstacles found and the degree of uncertainty the robot has about the current map.

There is unfortunately a lot of interdependence of each of the situation features on each other. The path information strength, for instance, depends on both the openness and the degree of exploration. To handle this interdependence, this feature can be calculated after the others are finished.

For example, the rules to determine the Openness are:

- If Degree of Allowable is *High* and the Map indicates the number of obstacles is *Very Low* then the Openness is *Very High*.
- If Degree of Allowable is *High* and the Map indicates the number of obstacles is *Medium* then the Openness is *Medium-High*.

These are only some of the rules. They require that the Allowable direction be analyzed to determine the percent of objects that the sensors see, and for the map to do a count of the number of obstacles in the area. Other rules for Degree of Exploration and Path Strength require this measure of Openness including more information from the map and the goals to get the final results.

Using Precompiled Changes

Once the situation has been evaluated by the situation recognition module, we need to output what changes are needed to best adapt to that situation. We use another fuzzy logic rule base to react to the recognized change in situation provided by the situation recognition module. This rule base uses the results to reason about what the values of the parameters should be based on previous tests that have been run on the robot in each of the given situations. At present the system returns the adaption values of three system parameters:

1. *Target Distance:* how far along the path should the target point be
2. *Nearness of Sensors:* what should be considered near? Are there different values for the forward and side sensors?
3. *Maximum Speed of the Robot:* This value is for both the value when going straight and for making turns.

The target distance depends on all three features. The amount of openness is a guide to how far the robot may safely look ahead. The stronger the path information strength the less the robot can look ahead. Finally, in exploration, the path is usually just a guideline that the robot should stay near.

How near obstacles are for the sensors determines the sensitivity of the sensors. If they are too sensitive, the robot cannot travel in enclosed places. If they are too weak, then the robot cannot travel at large speeds

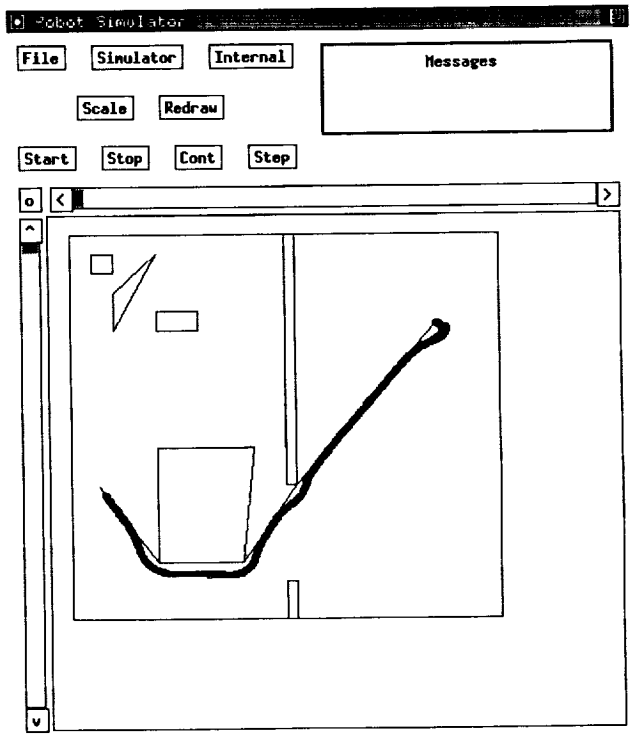

Figure 4: Outdoor Environment using Indoor Parameters

since it cannot sense obstacles soon enough to avoid them. This shows that the main overriding factor for nearness is the openness of the environment.

Finally the maximum speed of the robot is determined by combining the amount of openness with the goal of the robot. If the robot is exploring the robot will go slower to get more detailed scans. If the robot is traversing, it wants to go faster. If the robot is in an open area it can travel faster than in a less open area.

There is also some interaction between these parameter adjustments. The amount of nearness is influenced by the speed of the robot. The robot needs to be able to detect things further away as it goes faster. The robot also needs to project further ahead and anticipate corners better in open environments.

An example rule for generating an action is:

- If Environment is *Very Open* and Degree of Exploration is *Low* then Maximum Speed is *Very High*.
- If Maximum Speed is *Very High* and Degree of Exploration is *Low* then Target Distance is *Far* and Nearness is *Loose*.

The interdependance of the attributes can be seen in these rules.

Results

We have been working on implementing the situation recognition module. Figures 4-7 show two situations,

C-4

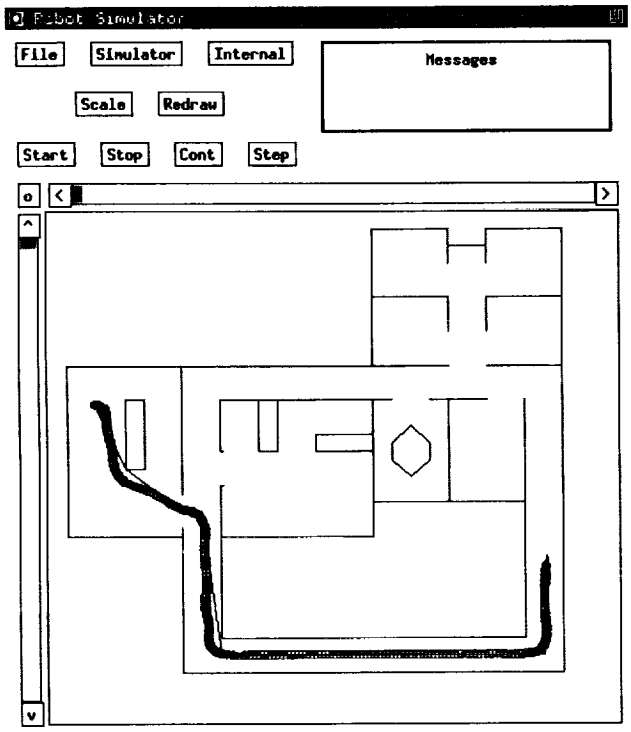

Figure 5: Indoor Environment using Indoor Parameters

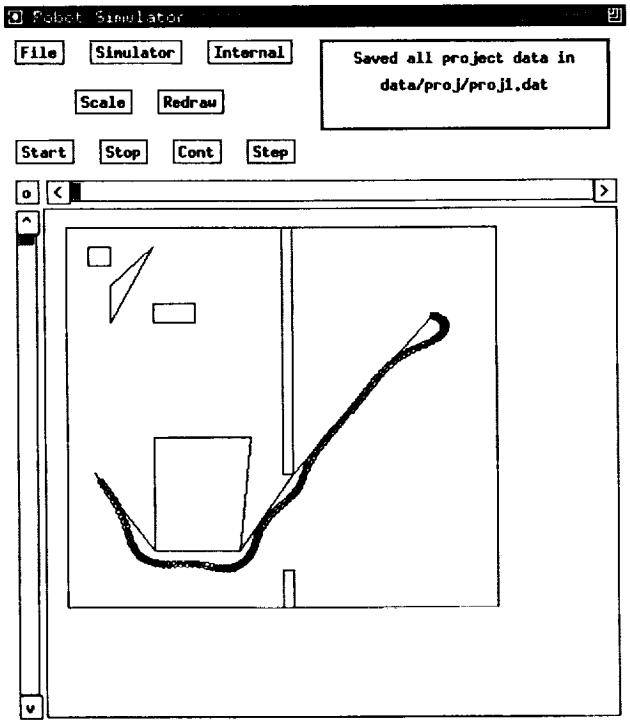

Figure 6: Outdoor Environment using Outdoor Parameters

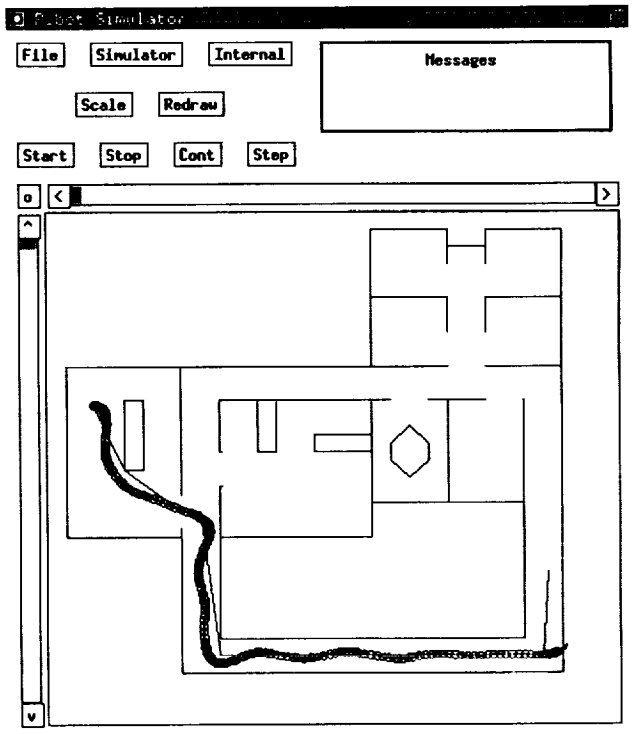

Figure 7: Indoor Environment using Outdoor Parameters

one an outdoor scene, the other using a floor plan of our lab. The figures show two different sets of parameters to show how changing the parameters can make the following of the path more efficient.

In Figures 4 and 5, the indoor parameters where used, meaning that the maximum speed was set to 10, the target distance to 50, and *nearness* to 30. The nearness is the point in Figure 1 where the membership stops being one and begins declining towards zero. In Figures 6 and 7, the outdoor parameters were used, i.e. the maximum speed was set to 17, the target distance to 70, and *nearness* to 60.

Important features of these figures include the fact that using the outdoor parameters, the indoor path could not be completed, while outdoors in a more open environment, the path may be slightly longer, but the time to complete the path was 150 steps, as opposed to 193 steps for the indoor paramenters, an improvement of 129 percent in the time needed to complete the task.

These runs were done using the same parameters throughout the run. In the future, the robot will be able to change parameters dynamically, based on the degree of openness and changing goals, as described in the section on situation recognition. This will allow the robot to be even more efficient, going faster when it can and slowing down in less open environments.

Adding Learning

We have recently been exploring the possibility of adding learning to the situation recognition and the action generation modules. Both of these modules use fuzzy logic rule bases to make decisions concerning the environment and how to react to it. While fuzzy logic rules are in general easy to create, there can be some problems.

The first problem is in scope. As the number of variables increase and the range of values they can take on becomes large, it becomes almost impossible to specify results for all possible combinations of values. The second problem occurs because rule bases are specified on the most important features of the problem. What if there are exceptions that can only be detected using variables not used in the fuzzy rule decision making process. Finally there is the problem that the rule base is created by an expert who is unsure of his reasoning and may give sub-optimal actions in a given situation.

We are currently working on learning in fuzzy rule based systems by using case based learning. The case based learning system adds cases to the rule base to overcome the problems created by using a static fuzzy rule base. We are researching ways to add cases so that their true motivation, whether it be as a novel case that should act as a rule itself, or as an exception which changes a small section of a rules scope.

Conclusions

As shown in the results section, there is a need to adjust the parameters based on the situation the mobile robot system is in. The current method of identifying the environment and determining the parameters is a start, but a more dynamic approach is needed. We are working on implementing the full system so that it can be run dynamically with the system. This will allow the robot to speed up in open areas while slowing down to safer speeds in more crowded environments.

References

[1] J. S. Albus. "Outline for a Theory of Intelligent". *IEEE Trans. Systems, Man, and Cybernetics*, 21(3):473–509, May/June 1990.

[2] B. Kosko. *Neural Networks and Fuzzy Systems*, volume 1. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1992.

[3] C.C. Lee. "Fuzzy Logic in Control Systems: Fuzzy Logic Controller – Part I". *IEEE Trans. Systems, Man, and Cybernetics*, 20:404–418, Mar 1990.

[4] C.C. Lee. "Fuzzy Logic in Control Systems: Fuzzy Logic Controller – Part II". *IEEE Trans. Systems, Man, and Cybernetics*, 20:419–435, Mar 1990.

[5] D. W. Payton. "An Architecture for Reflexive Autonomous Vehicle Control". In *IEEE Conference on Robotics and Automation*, pages 1838–1845, 1986.

[6] D.W. Payton, J.K. Rosenblatt, and D.M. Keirsey. "Plan Guided Reaction". *IEEE Trans. Systems, Man, and Cybernetics*, 20:1370–1382, Nov 1990.

[7] N. Pfluger, J. Yen, and R. Langari. A Defuzzification Strategy for a Fuzzy Logic Controller Employing Prohibitive Information in Command Formulation. In *Proceedings of the IEEE International Conference on Fuzzy Systems*, pages 717–723, San Diego, CA, March 1992.

[8] J. Yen and N. Pfluger. "Designing an Adaptive Path Execution System". In *Proceedings of the IEEE/SMC Conference*, pages 1459–1464, Charlottesville, VA, October 1991.

[9] J. Yen and N. Pfluger. "Path Planning and Execution Using Fuzzy Logic". In *Proceedings of the AIAA Conference on Guidance, Navigation, and Control*, pages 1691–1698, New Orleans, LA, August 1991.

[10] L. A. Zadeh. "Fuzzy Sets". *Inform. Contr.*, 8(3):338–353, June 1965.