A Prospective Fuzzy Logic approach to Knowledge-based Navigation of Mobile LEGO-Robot

Hrudaya Ku. Tripathy*[1], B.K.Tripathy[2] and Pradip K Das[3]
*[1]*Institute of Advanced Computer and Research, Prajukti Bihar, Rayagada-765002 (Orissa), India*
[2] *School of Computing Sciences, VIT University, Vellore-632014, Tamil Nadu, India*
[3] *Department of Computer Science & Engineering, Indian Institute of Technology Guwahati, North Guwahati-781039 (Assam), India*
hrudayakumar@hotmail.com, tripathybk@rediffmail.com, pkdas@iitg.ernet.in

Abstract

The development of techniques for knowledge-based navigation constitutes one of the major trends in the current research on mobile robotics. Fuzzy logic provides tools that are of potential interest to mobile robot control. Most applications of fuzzy logic in this field concern the use of fuzzy control techniques to implement individual behavior units. In this paper we discuss how fuzzy logic computation techniques have been used in the mobile Lego robot for knowledge-based navigation. Also we implemented a robot that learns how to avoid obstacle using online self-adaptation. The outputs from each behaviour's rule base are integrated using the command fusion process and made crisp using a modified defuzzification technique. The end result is very smooth motion control of the robot. The robot was built using the Lego RCX microcomputer and was designed with the subsumption architecture. We choose to implement the robot's brain using the Lego RCX due to the lower cost.

Keywords

Fuzzy Logic, Mobile Lego Robot, Navigation, Path finding, Obstacle avoidance.

1. Introduction

The development of techniques for robot navigation constitutes one of the major trends in the current research on mobile robotics. The goal of autonomous mobile robotics is to build physical systems that can move purposefully and without human intervention in unmodified environments that is, in real-world environments that have not been specifically engineered for the robot. The development of techniques for autonomous robot navigation constitutes one of the major trends in the current research on robotics. This trend is motivated by the current gap between the available technology and the new application demands. On the one hand, current industrial robots lack flexibility and autonomy.

In [1] typically, the robots perform pre-programmed sequences of operations in highly constrained environments, and are not able to operate in new environments or to face unexpected situations. On the other hand, there is a clear emerging market for truly autonomous robots. Possible applications include intelligent service robots for offices, hospitals, and factory floors; maintenance robots operating in hazardous or hardly accessible areas; domestic robots for cleaning or entertainment; semi autonomous vehicles for help to disabled people and so on. The problem of reliable navigation is the most pervasive in all of mobile robotics. For a robot to be truly mobile, it must be able to repeatable move from point to point while keeping track of its current location with respect to its environment and robustly recognizing when it achieves its goals.

In [2] this problem has received considerable attention; it still does not have a fully satisfactory solution: existing systems often lack flexibility, reliability, or both. Computational cost is also an issue. The main problem is that of dealing with uncertainty, which refers to the statistical distribution of errors in both control and sensing.

Mobile Robots currently employ a number of navigation strategies and use various sensors as navigational aids. In [3] the selection of sensors is directly dependent on the strategy the robot employs;

line following robots use vision systems to detect and follow the line, and track robots mount and remain on the tracks using specially designed wheels. Relative positioning robots rely on dead reckoning and error correction. Absolute positioning robots rely on landmark detection. However localization is a challenge for a robot that works in an unknown, uncertain, unpredictable and dynamic environment. The robot's sensor system has to perceive its environment and cope with imperfect and inaccurate sensor data.

As regards the navigation problem, some control algorithms based on artificial knowledge have been proposed. In [4], a globally stable control algorithm for wall following based on incremental encoders and one sonar sensor is developed. In [5], a theoretical model of a fuzzy based reactive controller for a mobile robot is developed. In [6], an ultrasonic sensor is used to steer an autonomous robot along a concrete path using its edged as a continuous landmark.

In [7], a mobile robot control law for corridor navigation and wall following based on sensor and odometric sensorial information is proposed. Recent research and application employing non-analytical methods of computing such as fuzzy logic, evolutionary computation, and neural networks have demonstrated the utility and potential of these paradigms for intelligent control of complex systems.

In the research proposed, we aim to demonstrate that fuzzy logic has features that allow an autonomously navigating LEGO robot to cope with the inherent uncertainties that occur when using sensor acquired location data as the navigational aid. In particular, fuzzy logic has proven to be a convenient tool for handling real world uncertainty and knowledge representation.

2. Related Studies

Few years ago, the Lego Company released a new range of Lego stuff called Lego Mindstorms System [8]. The goal of this new range was to give a practical (Lego) support for robotics development. With this kit you can build a Lego robot and command it from your PC. This new range uses the pieces of the Lego Techniques range, but Lego adds some special pieces:

- The Pbrick (Lego RCXtm Microcomputer): the main piece, it is a big Lego Brick, with a microprocessor (Hitachi H8/3292 micro-controller (16 MHz) with 16 KB ROM and 16 KB RAM) inside, and 3 input & 3 output ports on the top. It can communicate with the PC via an infrared port on the PBrick and infrared tower connected on the USB port of the PC.
- The output bricks: motors, lights.
- The input bricks (sensors): to detect contact, rotation, light and temperature.
- The cable bricks: it is just two small and simple Lego bricks with electrical contacts; it is used to connect a PBrick port to another special brick, for both inputs and outputs.

The robot we used consisted of 2 tracks wheels, which were able to move independently forward or backward. Mounted on the front were two touch sensors (left and right) and a light sensor with its face turned to the ground. All these elements are shown in figure. 1.

We made use of these Lego parts for our body design as in fig. 2,3&4. We made use of Lego because it provides a very quick way of prototyping the physical body of the robot and allows us to modify our design with ease. The problem with Lego parts is that they come apart very easily and this is not desirable, especially if the robot is supposed to collide into obstacles (to learn how to avoid them). We interlocked the various parts of the robot's body so that the parts do not fall off easily. This inevitably adds weight to the robot's body and hence reduces its angular velocity and speed.

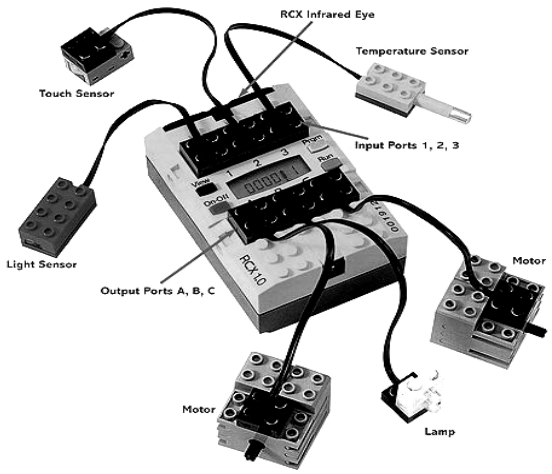

Figure 1. Fundamental elements of LEGO Robot[8].

Our programs are downloaded to the RCX via an infrared Tower, which is connected to the PC's USB port. The RCX has an in-built infrared transmitter and receiver for the purpose of communicating with the IR tower and other RCX bricks. The RCX can be programmed using a variety of languages, like the C-

like Not-Quite-C (NQC) [9], C/C++ on legOS [10] and Java on leJOS [11].

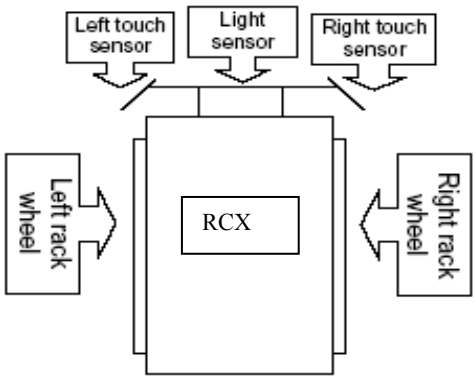

Figure 2. Schematic of the Lego Robot we used.

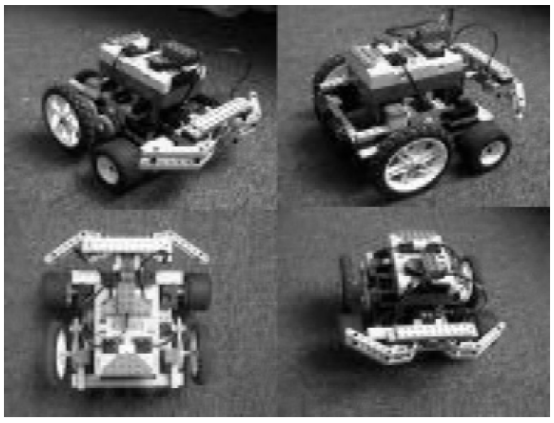

Figure 3. Physical structure of our Intelligent Lego Robot.

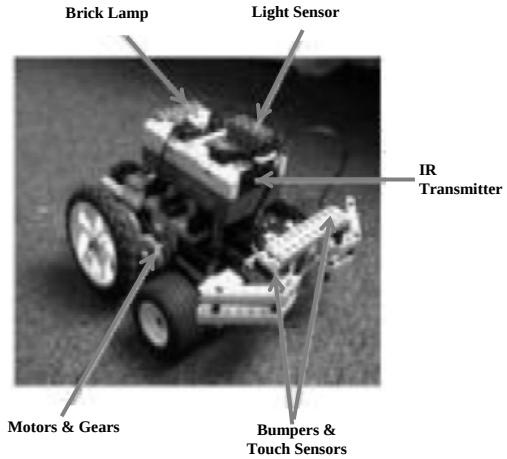

Figure 4. The Robot: Sensors & Actuators

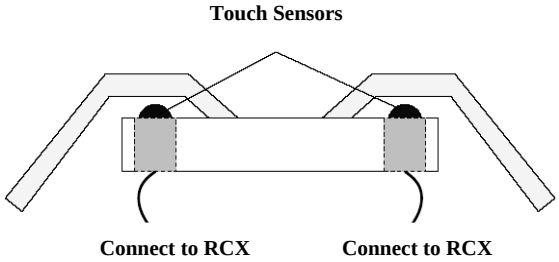

Figure 5. Bumpers & Touch Sensors

3. Fuzzy Logic

Originally advocated by Zadeh [12] and Mamdani and Assilian [13], fuzzy logic has become a means of collecting human knowledge and experience and dealing with uncertainties in the control process. Now fuzzy logic is becoming a very popular topic in control engineering. Considerable research and applications of this new area for control systems have taken place. Control is by far the most useful application of fuzzy logic theory, but its successful applications to a variety of consumer products and industrial systems have helped to attract growing attention and interest. Based on its design simplicity, its ease of implementation, and its robustness properties, a fuzzy logic controller is used in this paper to control the navigation behaviour of an autonomous mobile robot.

3.1 Fuzzy Behavior Controller

The navigation system employed on our robot is based upon a fuzzy logic control system. Both the path following and obstacle avoidance behaviour are implemented using a fuzzy logic based architecture. Each behaviour consists of a set of fuzzy control rules and a fuzzy inference module. The output from each behaviour is not be a crisp control value, but is instead a fuzzy set. These sets are then combined through a command fusion module and defuzzified to produce a crisp output value figure 6. The output of the path following behaviour produces a fuzzy set representing the desired turning direction while the output from the obstacle avoidance behaviour produces a fuzzy set representing the disallowed turning directions. The method has only been tested in simulation, but has produced very favorable results. Fuzzy logic control techniques can be used to design individual behavior units. Fuzzy logic controllers incorporate heuristic control knowledge in the form of if-then rules, and are a convenient choice when a precise linear model of the system to be controlled cannot be easily obtained.

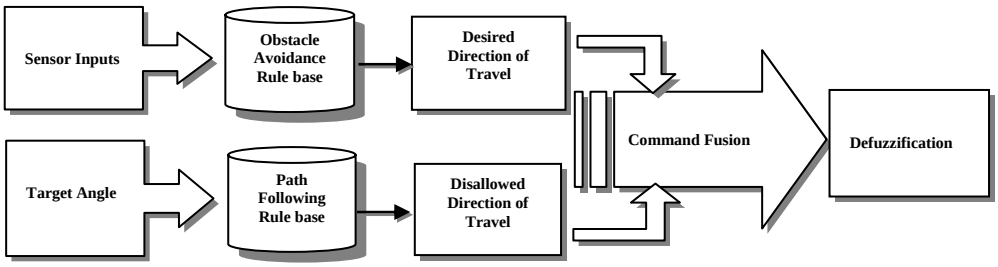

Figure 6. A fuzzy logic based navigation controller.

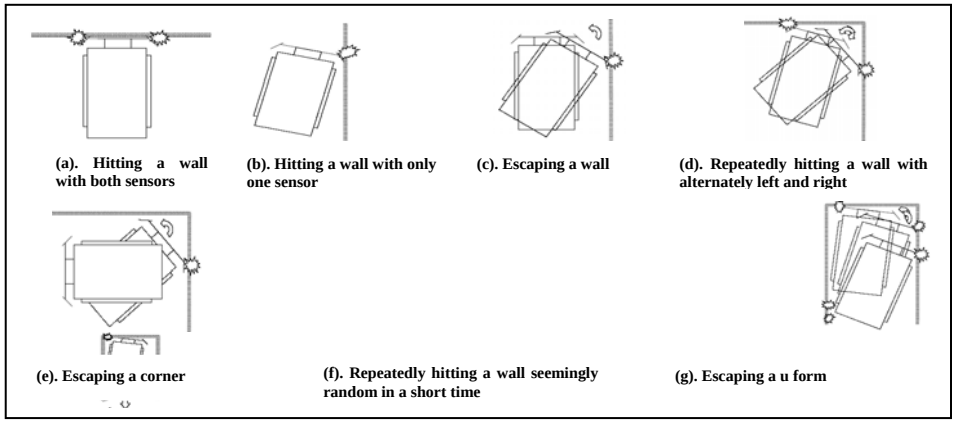

Figure 7. Recognition of different events/situations by Lego robot.

Fuzzy logic controllers have also demonstrated a good degree of robustness in face of large variability and uncertainty in the parameters. We have developed a number of robust behaviors using the techniques of fuzzy control. These include purely reactive behaviors, like "Avoid-Obstacles", and goal-achieving behaviors, like "Follow-path". Specifically, we defined 5 simple event-action conditions, i.e. with which action the robot responds when it recognizes some event/situation represent in figure 7. They are:

1. No event
2. Hitting a wall straight
3. Hitting a wall sideways
4. Stuck in a corner
5. Stuck in a U form figure

We have developed a fuzzy localization technique to compare map and perceptual information as per above in order to re-localize the robot during navigation. The peculiar feature of this technique is that we use fuzzy sets to represent the approximate position of objects. We have developed an arbitration mechanism based on fuzzy logic. For instance we can use a perceptual condition to decide between two alternative behaviors and the following rules can be used to navigate to a target while reactively avoiding obstacles on the way:

IF NOT (obstacle) THEN Follow-path/Go-To-Target.
IF obstacle THEN Avoid-Obstacles
IF obstacle-close-right AND NOT(obstacle-left) THEN turnsharp-left

When the obstacle is only partially close, both behaviors are partially activated; thus, the commands issued by the Follow-path/Go-To-Target behaviour can be taken into account during obstacle4 avoidance maneuvers. This is an important feature for reactive navigation. We can also sequence two behaviors B_1 and B_2 aimed at two goals G_1 and G_2 by using context rules of the form,

IF G1 not achieved THEN B1
IF G1 achieved THEN B2

Similarly, we implemented the fuzzy rules for follow behaviour are as follows,

IF (path-too-right $\wedge$ $\neg$path-angled-left) THEN turn-medium-right
IF (path-too-left $\wedge$ $\neg$path-angled-right) THEN turn-medium-left
IF (path-angled-right $\wedge$ $\neg$centreline-left) THEN turn-smooth-right
IF (path-angled-left $\wedge$ $\neg$centreline-right) THEN turn-smooth-left

The condition-action rules were implemented using simple if-then statements. The order of the statements is thereby important.

4. Experimental Procedure & Results

The controller of the robot consisted of a so-called RCX unit, which can be programmed to set the motors and read the sensors. We used the NQC1 programming language. This is basically a limited kind of C dialect. Limitations include having only 31 variables, no arrays and no nested function calls. However the language also features an extension to C in the form of parallel running tasks. We like to mention that there were alternative languages available (with accompanying OS') like legOS and pbForth, but we did not use them.

We make use of two-9V DC servomotor to drive the robot. The two motors are connected to the first (A) and third (C) output ports provided in the RCX and the program will write into the registers of the output ports to control the speed of the motor. We measured the various parameters of the motor using 6 new 9V batteries and they are shown in Table 1 to 3. As we can see from the values in these tables, the linear and angular distances traveled are proportional to the time that the motor runs (with a small initial start-up cost). One end of the touch sensors (simple switch) are connected to the bumpers of the robot and the other end are connected to the input ports (1 and 3) of the RCX to provide collision detection as shown in Figure 5. Whenever the bumper collides with an obstacle, it will cause the touch sensors to be pressed, which in turn generates an interrupt at the microcontroller. The program reacts to this interrupt with a collision resolution behaviour that moves the robot away from the obstacle. The bumpers were built in such a way that its width is almost as wide as the robot's body so that it can successfully detect any collision when it occurs.
The IR transmitter will broadcast packets and the light sensor will measure the difference between two consecutive readings to determine if there is an obstacle in front of the robot. The light sensor is very sensitive to infra-red signals, hence when the infra-red packet sent by the IR transmitter gets reflected back by an obstacle in the path of the robot, it will cause a jump in the light sensor's readings.

The NQC code for performing this operation is shown below. A proximity is detected by an abrupt change to two consecutive light sensor readings, indicating that the IR packets sent out by the *send_signal()* task has been reflected back. We place the light sensor slight above the IR transmitter in our robot so that the outgoing IR packets do not interfere with the readings of the light sensor. We also ensure that the IR transmitter is not blocked by other parts of the robot causing the proximity sensor to give erroneous results.

Due to the hardware limitations it is not possible to achieve good robot control results by doing the fuzzy calculations on board of RCX box of the Lego robot, thus the response of the system in real time is too slow to allow the robot to attain a steady state.

4.1 Program Code (NQC)

```
task send_signal() {
SetPriority(MIN_PRIORITY-2);
while(true) {
SendMessage(0); /* Sends IR packets,
takes approx 30 ms*/
   Wait (20); /* Wait for 100 ms to allow
detection */
  }}
task avoid_obstacle() {
  SetPriority(MIN_PRIORITY-3);
  int previous_proximity;
  int diff;
  start send_signal;
while ( true ) { /* Calculate the
difference between two
consecutive readings of the light sensor
*/previous_proximity = LIGHT_SENSOR;
  diff = LIGHT_SENSOR -
previous_proximity;
  diff = ( diff < 0 ) ? -diff : diff;
if ( diff > proximity_threshold ) {
  acquire(ACQUIRE_OUT_A + ACQUIRE_OUT_C)
{move_backward();
  turn_right();}}}}
```

Since the robot is supposed to act as an autonomous agent, it does not interact with the user. As such it does not feature a user interface and we could therefore not give much attention to usability. The values read from the light sensor when it passes an object are hard coded in the source. They are of course only valid when the same lighting is present as when we calibrated the code to them. The functionality of our robot depends on many factors: the setting must be lighted the right way, and the batteries need to be at full power. If they are weak, the robot moves slower and our turns are not as effective. A whole lot of other factors are beyond our control. As mentioned in this report, they include slipping the wheels, hitting the walls without detecting a hit, inaccuracy of the movements etc.

5. Conclusion

Fuzzy logic has features that are particularly attractive in light of the problems posed by mobile robot navigation. Fuzzy logic allows us to model different types of uncertainty and imprecision to build robust controllers starting from heuristic and qualitative models. In this paper knowledge based

fuzzy logic controller to control the motion of differential drive mobile Lego robots has been presented. Simulations were carried out on a mobile Mindstrome Lego robot to test the performance of the proposed fuzzy controller.

Finally, we employed fuzzy techniques to represent and use approximate maps. Most of the approaches to representing spatial uncertainty in robotics are based on probabilistic techniques. These techniques are adequate when: (i) the underlying uncertainty can be given a probabilistic interpretation; and (ii) the probabilistic data required by these techniques areavailable. Both conditions may fail for the autonomous robots; in these cases, our solution may offer a valuable knowledge based alternative. Our experiments have shown that fuzzy behaviors are able to operate relying only on approximate maps and imprecise sensing a most important requirement for an autonomous mobile robot intended for use in unstructured environments. The end conclusion is, we're not really trying to specifically find the exit, but just programming the robot to move randomly and hope its movements cover as much of the world as possible.

Table 1. Angle Turned Versus Time of Turning

Angle Turned	45°	90°	180°	270°	360°
Time of Turning (secs)	1.0	1.9	4.1	6.1	8.2

Table 2. Distance Traveled Versus Time of Traveling

Time of Travel (secs)	0.5	1.0	1.5	2.0	2.5
Distance Traveled (mm)	132	267	378	516	653

Table 3. Distance of Obstacle Versus Proximity Sensor Reading

Distance for Obstacle (mm)	0	127	254	381	508
Proximity Sensor Readings	222	128	83	41	16

6. References

[1] A. Saffiotti, “The uses of fuzzy logic in autonomous robot navigation”, Soft Computing 1(4), pp 180-197, 1997.

[2] Craig Becker, Joaquin Salas, Kentaro Tokusei, and Jean-Claude Latornbe, “Reliable Navigation Using Landmarks”, IEEE lnternatlonal Conference on Robotics and Automation, pp 401-406, 1995.

[3] Shérine M. Antoun Phillip J. McKerrow, “Landmark Navigation with Fuzzy Logic”, Proceedings of the Australasian Conference on Robotics & Automation, December 6 - 8, 2006, Auckland, New Zealand.

[4] R. Benporad, M. Di Marco, A. Tesi, “Wall-following controllers for sonar-based mobile robots”, IEEE Conference on Decision and Control, San Diego, December 1997.

[5] Ole Jakob Sordalen, “Feedback Control of Nonholonomic Mobile Robots”, doctoral thesis, Department of Engineering Cybernetics, The Norwegian Institute of Technology, 1993.

[6] Danny Ratner, Phillip McKerrow, “Navigating an outdoor robot along continuous landmarks with ultrasonic sensing,” Robotics and Autonomous Systems 45, pp. 73–82, 2003.

[7] Ricardo Carelli Eduardo Oliveira Freire, “Corridor navigation and wall-following stable control for sonar-based mobile robots,” Robotics and Autonomous Systems 45, pp. 235–247, 2003.

[8] Baum, David, “Definitive Guide to LEGO MINDSTORMS” Second Edition. New York, Apress 2003.

[9] Not-Quite-C. http://www.enteract.com/ dbaum/nqc/.

[10] LegOS. http://legos.sourceforge.net/.

[11] LeJOS. http://lejos.sourceforge.net/.

[12] L.A. Zadeh, “Fuzzy sets, Information and Control”, pp. 338–353, 1965.

[13] E.H. Mamdani, S. Assilian, “Application of fuzzy algorithms for control of simple dynamic plant”, Proceedings of the Institute of Electrical Engineers 121, pp. 1585–1588, 1974.

Authors’ bio.

Hrudaya Ku. Tripathy is presently working as Asst. Professor in the Dept. of Computer Science & Engineering at Institute of Advanced Computer & Research, Rayagada, Orissa, India. He has 10 years of teaching experience in UG/PG courses. He has M.Tech from Indian Institute of Technology Guwahati, and pursuing his Doctorate under Berhampur University, Berhampur Orissa, India.

B.K.Tripathy is presently working as Professor in the School of Computing Science at VIT University, Vellore, India. Having around 27 years of teaching experience in UG/PG courses. He has M.Tech from University of Poona, Pune, India. and completed his PhD. He has published more than 50 research papers in national and international journals.

Pradip K. Das is presently working as Asst. Professor in the Dept. of Computer Science & Engineering at Indian Institute of Technology Guwahati. Having around 20 years of teaching experience in both UG/PG classes. He has M.Sc from Delhi University and completed his PhD. He has published more than 30 research papers in national and international journals.