Windows Application for Intelligent Mobile Robot Control

Monica Dragoicea

Nicolae Constantin

University "Politehnica" Bucharest

Faculty of Control and Computers

Automatic Control and System Engineering Department

Splaiul Independentei 313, 77206 – Bucharest

ROMANIA

Abstract: This paper introduces a new integrated open-architecture platform for mobile robot simulation, controller design and real-time implementation - MobLib. The proposed system provides not only a user-friendly simulation and educational platform but also an open-architecture, control system development environment for real-time control of mobile robot systems. Besides the built-in control algorithms, this environment also enables the user to integrate user specified robots, control algorithms and trajectory planning schemes into the system. The focus of this work was on the application of fuzzy logic techniques to the design and implementation of basic navigation behaviors, and to the combination of basic behaviors to form complex behaviors to execute full navigational plans. For this, suitable simulation and real-time evolution environments are needed.

Keywords: real-time control of complex systems, fuzzy control, mobile robots

Monica Dragoicea received the B.S. and Ph.D. degrees from University Politehnica of Bucharest, Romania, in 1993 and 2000, in Electrical Engineering and the M.S. degree from Oakland University, Michigan, USA, in 1999, in Engineering Management.She was the recipient of the 2000 Werner von Siemens Excellence Award for the best Ph.D. thesis on Automatic Control. She is currently Lecturer on Automatic Control and Systems Engineering Dept. at University Politehnica of Bucharest. She is author of 4 books and more than 20 papers in intelligent techniques for mobile robots area. She is a member of SRAIT - the Romanian Society of Automatic Control and Technical Informatics.

Nicolae Constantin was born at Ploiesti, Romania on 17th Nov., 1963. He received the MSc degree in Control Systems and Computer Science from Polytechnical Institute of Bucharest, in 1988. In 1997 he received PhD degree in Control Systems at University Politehnica of Bucharest. Since 2000 he has been Associate Professor at the Automatic Control and Systems Engineering Dept. at University Politehnica of Bucharest. His current research interest include robust control, adaptive control, hybrid intelligent techniques for nonlinear control. He is author of 3 books and more than 40 papers in control engineering area. She is a member of SRAIT - the Romanian Society of Automatic Control and Technical Informatics.

1. Introduction

Engineering and computer science are core elements of mobile robotics, obviously, but when questions of intelligent behavior arise, artificial intelligence, cognitive science, psychology and philosophy offer hypothesis and answers.

An *intelligent robot* is a mechanical creature which can function autonomously. "Intelligent" implies that the robot does not do things in a mindless, repetitive way; it is the opposite of the connotation from factory automation. The "mechanical creature" portion of the definition is an acknowledgment of the fact that our scientific technology uses mechanical building blocks, not biological components. It also emphasizes that a robot is not the same as a computer. A robot may use a computer as a building block, equivalent to a nervous system or brain, but the robot is able to interact with its world: move around, change it, etc. A computer doesn't move around under its own power. "Function autonomously" indicates that the robot can operate, self-contained, under all reasonable conditions without requiring recourse to a human operator. Autonomy means that a robot can adapt to changes in its environment or to itself (e.g. when a part breaks) and continue to reach its goal (Dragoicea, 2000).

The purpose of this paper is to design *an heuristic control environment* for a nonlinear process (the mobile robot being in a tight interaction with the outer environment) with a variable behavior structure. Adaptive behaviors are defined and implemented in a complex informatic structure.

This paper introduces a new integrated open-architecture platform for mobile robot simulation, controller design and real-time implementation - MobLib. The proposed system provides not only a user-friendly simulation and educational platform but also an open-architecture, control system development environment for real-time control of mobile robot systems. Besides the built-in control algorithms, this environment also enables the user to integrate user specified robots, control algorithms and trajectory planning schemes into the system. The software structure of the application is depicted in section 4. Section 5 presents case studies and results obtained by conducting real-time control sessions on a Khepera mobile robot. Section 6 gives some conclusions and directions for further research.

2. Fuzzy Multi-behavioral Control

As extensions of living creatures like-intelligence, behavior-based approaches to mobile robot control have gained increasing popularity, being supported by considerations arising from the study of animal behavior, by architectural concerns (Brooks, R., 1986) and by their implementation conveniences.

The behavior is the fundamental building block of natural intelligence. A "behavior" is a mapping of a sensory inputs to a pattern of motor actions which then are used to achieve a task. The reactive paradigm makes extensive use of reflexive behaviors (i.e. stimulus - response behaviors or S - R behaviors), to the point that some architectures only call a robot behavior a behavior if it is of S - R type.

The agent can retain its relationship to the environment by using a set of behaviors each of which maintains a mapping from sensory information to some control parameters for actuators. Therefore, multiple models can be used to describe the different environment situations and the control is further affected by switching to an appropriate controller (Dragoicea, 2003), see figure 1.

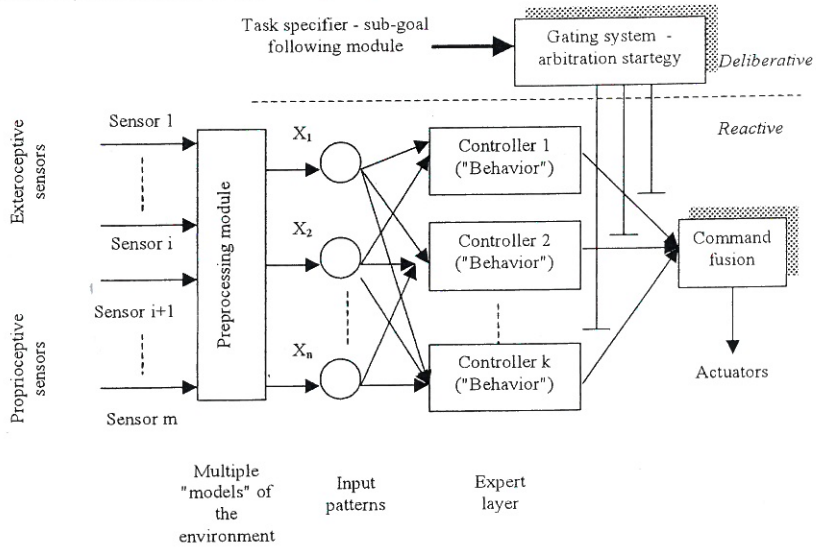


Figure 1: Multi-behavior Based Controller for Autonomous Navigation

A "model" of the environment represents a sensory situation (representing stimuli) that will be transformed into an input pattern, X_i, for the expert layer. Each controller from the expert layer is responsible for the implementation of a certain "behavior", i.e. a mapping from the sensory information to some control parameters for the actuators. As shown in figure 1, the minimum representation required in this approach consists of two parts: sensory situations representing stimuli, and situation to reaction mapping.

Following results obtained in (Dumitrache, 2002) and (Dragoicea, 2003), the reactive behaviors are all implemented here as fuzzy logic controllers (FLC). As can be seen in figure 2, the input and output variables to the controller are identified to be the sensory inputs (in each of the four abstracted directions), respectively, the left and right motor controls (or wheels velocities). The general architecture of the FLC that is used to control the robot's motion is designed as shown in figure 2.

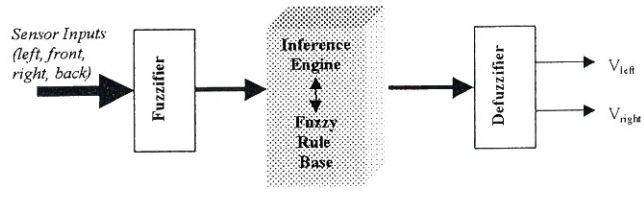


Figure 2: The Architecture of the FLC Used for Elementary Behavior Implementation

Considering the strategy presented in (Dragoicea, 2003), this work proposes a new approach for designing a *fuzzy reactive multi-control system* for the *real-time autonomous navigation of mobile robots*, in which each behavior is realized as a separate FLC. The multiple outputs are merged together by a gating system acting as a supervisory module (under a certain arbitration strategy). In this way, it is possible that simpler behaviors are combined in order to obtain more complex navigation behaviors.

3. Real-time Control of the Mobile Robots

To date, there are already available some freeware and commercial software for mobile robot control, such as the LabView control interface for the Khepera mobile robots, Kiks toolbox as a companion package for MATLAB, etc. For both of them, only a collection of functions is provided to be used in either LabView or MATLAB environments. Although Kiks / MATLAB and LabView provide powerful visualization tools of the mobile robot actions in the environment, they do not have the functionality to integrate real-time control algorithms, dynamic simulation and trajectory planning together.

The application is developed using the standard style of an Windows application that can be run on any Win32 platform: Windows95/98/NT. Figure 3 shows the main window platform of the open-architecture for mobile robot real-time control.

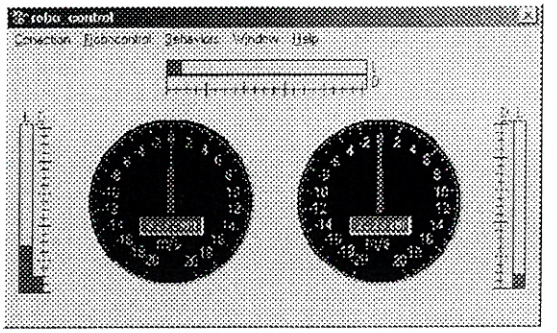
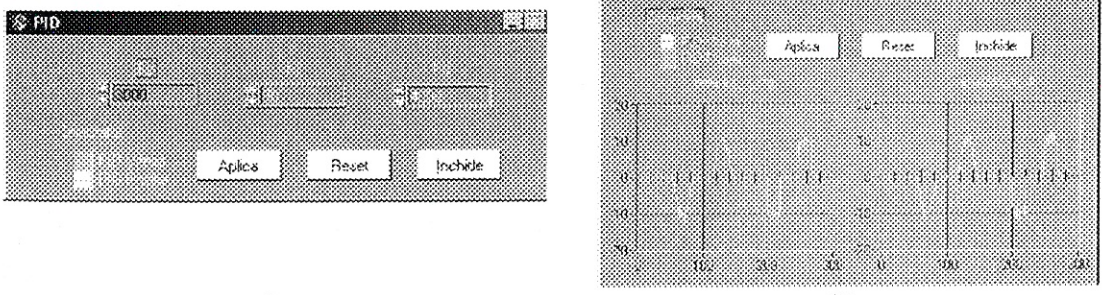


Figure 3: Sample Layout of the System Platform

The main window of the application is divided into two panes. The upper one is the *System Configuration* pane in which all the configuration settings are managed while the lower one is the *Performance Graphs* pane that can be used to display the control performance graphically.

A friendly and interactive user interface is provided in the *System Configuration* pane, which is divided into five functional modules to represent the five main application menus:

- *Connection*. The connection through the serial link with the robot can be opened / closed using this menu (*Connection / Open*, *Connection / Close*). The graphical window of this menu allows settings for the serial communication (port number, communication speed, number of the information bits, number of the stop bits);
- *Robocontrol*. Maintains a list of control functions for the mobile robot. *Configurare/PID* (figure 4) allows the parameter configuration for the two PID controllers (for position and speed), while *Configurare/SpeedProfile* (figure 5) allows to configure the velocity profile for the mobile robot. The *SetSpeed* function (figure 6) is used to control the velocities of the two CD motors of the wheels, in real-time (there are two sliders and two numerical controls available, one for each wheel). The *Encoders* function (figure 7) allows the visualization of the content of the two encoders for the wheels. The *Sensors* function (figure 8) will open a new window that allows to visualize the measurements of the eight infra-red proximity sensors, whose values are actualized at equal times.



a) b)

Figure 4: a) Interface for PID Parameters Setting,
b) Interface for PID Parameters Setting and Reference Tracking Visualization

- *Behaviors*. This menu allows the user choice between different behaviors of the robot (i.e Braitenberg vehicle behavior, reactive behaviors, see (Dragoicea, 2003)). Different other behaviors can be easily implemented here, extending the overall functionality according the user's expectations. While running according to a certain behavior, the sensors indications are available for visualization (e.g. by using the sub-menu *Robocontrol/Sensors*) as well as encoders evolution (through the *Robocontrol/Encoders* sub-menu).

In this way, the MobLib enables a user to customize his own robot and build it into the system for simulation and real-time control. Different tracking performances are displayed with different visualization tools in the lower *Performance Graphs* pane. For example, in this side of the window there are indicators to display left, right and front sensors values (light and distance). In the center there are two circle indicators of the wheels velocities.

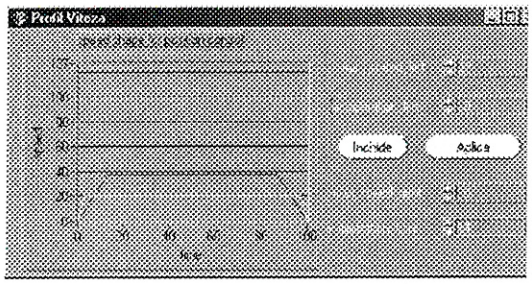


Figure 5: All the Parameters of the Speed Profile can be Controlled

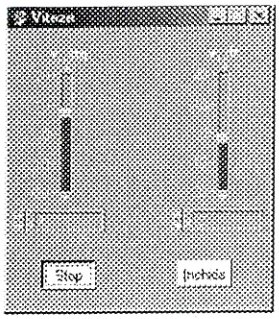


Figure 6: Motors Panel: 2 Sliders Controlling the Speed of Each Wheel

4. The Structure of the Application

In order to make the system more flexible, portable, and a "true" open-architecture platform for real-time control of the mobile robots, the DLL technology has been incorporated into the system.

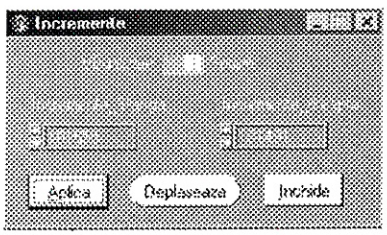


Figure 7: Control and Visualization of the Encoders

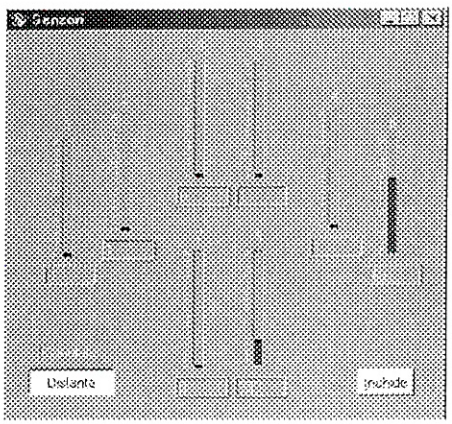


Figure 8: Sensors Panel: 8 Gauges Displaying the Infrared Values

The application for real-time control of the mobile robots has a modular structure (figure 9):

- the serial communication module (*RoboSeriala*) is responsible for the communication between the computer and the real robot;
- the graphical interface module (*RoboInterfata*) implements the Windows application elements for the communication between the user and the robot;
- the module *RoboVirtual* communicates with the real robot at regular time intervals, in order to gather information like wheels velocities, sensors values, that will be stored in a buffer and to send the control signal to the robot. The role of this module is to prevent two interface elements to read / write on the serial link at the same time;
- the *Braintenberg vehicle* control engine;
- the *FuzzyClasses* module implements the fuzzy control engine where different reactive behaviors are defined.

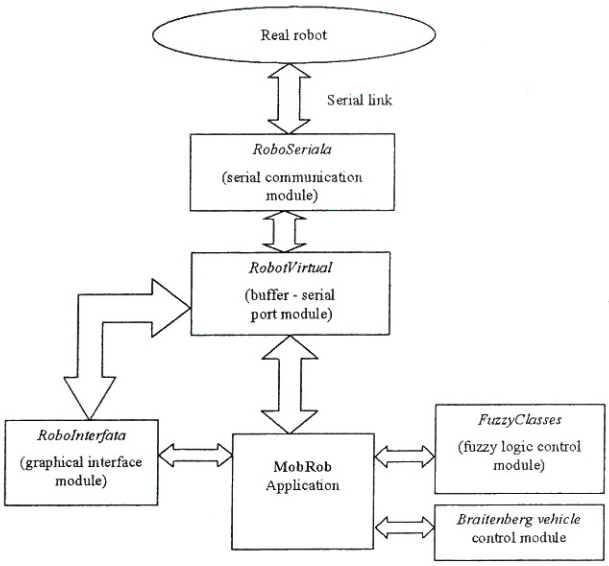


Figure 9: The Block Structure of the Application

4.1 The serial Communication Module

This module is a DLL library that contains specific communication functions for the Khepera mobile robot. The serial communication protocol allows the complete control of the functionalities of the robot through an RS232 serial line. To control the functionalities of the Khepera robot (motors, sensors, etc), a set of commands are implemented in the control protocol. Also in this case, the communication with the Khepera robot is made by sending and receiving ASCII messages. Every interaction between the host computer and Khepera is composed by:

- a command, beginning with one or two ASCII capital letters and followed, if necessary, by numerical or literal parameters separated by a comma and terminated by a carriage return or a line feed, sent by the host computer to the Khepera robot;
- a response, beginning with the same one or two ASCII letters of the command but in lower-case and followed, if necessary, by numerical or literal parameters separated by a comma and terminated by a carriage return and a line feed, sent by the Khepera to the host computer.

For example, to set the speed of the two motors the following format is used: "D, speed_motor_left, speed_motor_right\r".

4.2 The Virtual Robot Module

The *RobotVirtual* module is a buffer between the real robot, from which data are read and to which control signals are sent through the serial link, and the other modules that use the data. In this way, the real-time control is much easier to accomplish, the number of read/write operations to the serial ports being reduced. At the same time, the different applications that uses the data contained in this module are synchronized. The *RobotVirtual* module is implemented as a DLL library that creates, at the beginning of a control session, a memory area that will be shared by all the modules of the application.

The data contained by this buffer is updated by a thread that at regular time intervals calls an internal function of the library (*RoboActualizare*). This function reads data from the real robot and writes to the serial link the control signals. The way in which this memory buffer is created requires a strong synchronization between the modules of the application, for mutual exclusion. In order to avoid the simultaneous reading and writing, the reading thread will be suspended till the end of the writing operation. The thread that makes the actualization of the data in the buffer is pre-planned on time condition.

4.3 The Fuzzy Logic Control Module

The *FuzzyClasses* module is the engine that implements the reactive behaviors as fuzzy logic controllers. It represents a C++ fuzzy classes library used for the implementation of the fuzzy controller class (*CfuzzyRegulator*), following the method described in (Dragoicea, 2003). The fuzzy controller used here (figure 10) in order to define fuzzy reactive behaviors has two sets of inputs, the crisp values of the input variables and the linguistic variables associated with these inputs.

The *CInOut* class was specially designed in order to store the numerical value of the input at a certain point in time (m_valCrt), as well as the min (m_valMin) and max (m_valMax) values of this variable.

The *CAlfabetFuzzy* class was created in order to represent the linguistic variables, and it contains a list of associated linguistic terms. These linguistic terms are represented by the *CSimbolFuzzy* class, that actually implements the membership functions.

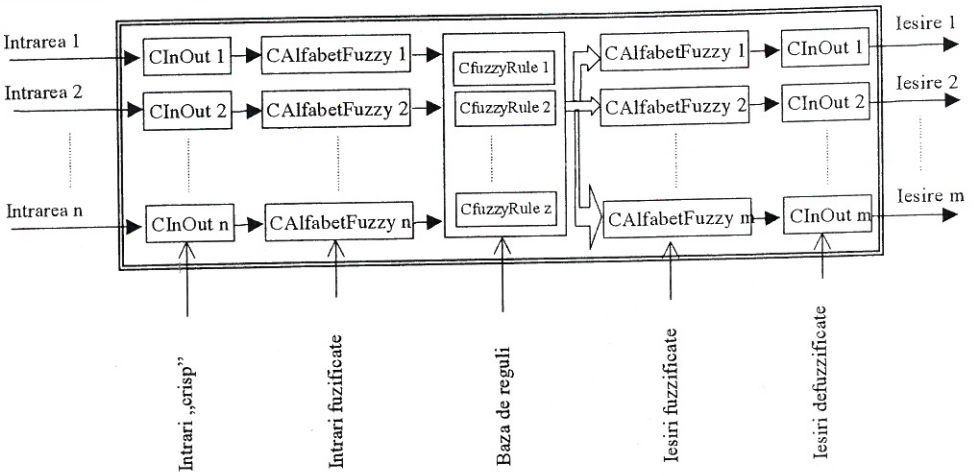


Figure 10: The Block Structure of a Fuzzy Controller for Fuzzy Behavior Implementation

When the crisp information is presented to the fuzzy controller, it will be firstly presented to the *CInOut* object and than it will be fuzzified in a *CAlfabetFuzzy* class. The fuzzified inputs are used for the interpretation of the rules.

The rules are clauses of the following type:

IF (context) **THEN** (command)
that means

IF (In 1 is TermLingv1 and ... and In n is TermLingv2)

THEN (Out 1 is TermLingv3 and ... and Out 2 is TermLing4)

and are implemented in a *CFuzzyRule* class. The rule base of the fuzzy controller, that is implemented in the *CBazaReguliFuzzy* class, is in fact a list of such rules together with specific operations for the maintenance and manipulation of the rules.

The outputs of the fuzzy controller are implemented in the same way as the inputs, the only difference being that of the order in which the operations are performed: the information flow goes first through the *CAlfabetFuzzy* objects and than through *CInOut* object for defuzzification. Figure 11 presents the classes hierarchy that effectively implements the fuzzy controller class. For the multi-input, multi-output fuzzy controllers, the controller contains lists of inputs / outputs of the *CInOut* type, and lists of linguistic variables of *CAlfabetFuzzy* type, associated to the to the *CInOut* inputs / outputs.

5. Case Studies

In order to demonstrate the functionalities of the MobRob and have a clear picture of the overall performance, several case studies have been developed. All these cases were developed following the companion results in (Dragoicea, 2003) and clearly proves in real-time the viability of the multi-behavioral fuzzy control strategy for autonomous navigation of the mobile robots.

By using fuzzy logic different complex behaviors can be implemented in real-time. Among these are wall following behavior, corridor following, goal following behavior, collision avoidance, goal following with collision avoidance behavior, etc.

As an example, table 1 shows practical rules that implement a wall following behavior on our robot test-bed, the mobile robot Khepera.

Table 1: Fuzzy Rules for a Wall Following Behavior

IF (dist_front_OK $\wedge$ dist_left_far) THEN turn_medium_left
IF (dist_front_OK $\wedge$ dist_left_OK) THEN turn_smooth_left
IF (dist_front_small $\wedge$ dist_left_medium) THEN turn_smooth_right

Multiple behaviors may be combined by context-blending. Context-dependent blending may be implemented in a hierarchical fuzzy controller, as shown in figure 1. Here, the multiple outputs are merged together by a gating system acting as a supervisory module (under a certain arbitration strategy).

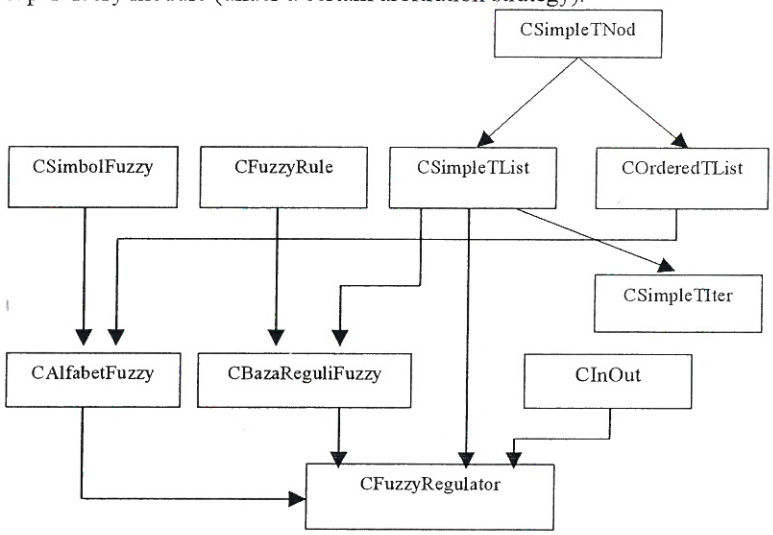


Figure 11: Fuzzy Classes Hierarchy

Following the above presented methodology for fuzzy controllers development, different fuzzy behaviors are developed here for:

- *position tracking*. Position tracking is a very simple behavior that can be used together with other elementary behaviors in order to obtain more complex control strategies for autonomous navigation;
- *wall following*. The wall following behavior is a tracking control mechanism able to follow any continuous surface at some fixed distance from contact, a feature that is most useful when mapping an unknown environment.;
- *collision avoidance*. Collision avoidance is a behavior that can be implemented by using the two other behaviors that where presented above.

Figure 12 shows a simple example of context-dependent blending where two different behaviors are employed to follow a corridor (FOLLOW) and to go to a target position (TRACKING). In this way, it is possible to implement a the position tracking behavior while avoiding the obstacles that can appear on the way to the target.

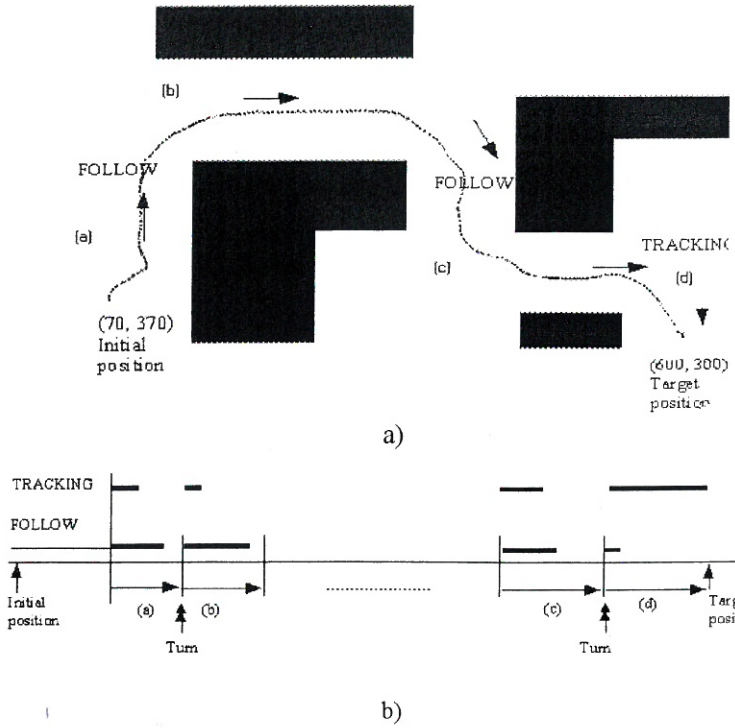


Figure 12: Collision Avoidance Behavior - as Blending of wWall (corridor) Following and Position Pracking

The COLLISION AVOIDANCE behavior is built by blending two basic behaviors:

- the wall following behavior (FOLLOW) keeps a convenient cruising speed while the way is clear of obstacles. As mentioned before (see also (Dragoicea, 2003)) this is a compound behavior, consisting of two elementary behaviors for left and right wall following;
- the position tracking behavior (TRACKING) keeps the robot on the way to the target by continuously evaluating the distance to the target (Dragoicea, 2003).

All these behaviors are purely reactive, simply monitoring proximity sensor readings to determine the presence of obstacles, as well as encoders turns in order to determine the current position of the robot.

The meta-rules used to encode the context in this example are of the following type:

IF approaching_obstacle THEN apply FOLLOW
IF NOT(approaching_obstacle) THEN apply TRACKING

ceTherefore, this complex behavior uses the wall following behavior in order to follow the contour of the obstacle. The robot goes to the target according to the position tracking behavior; when an obstacle appears on its path to the target, the mediator will make the switch to the wall following behavior, the shape of the obstacle is followed till the obstacle remains behind the robot, then the way to the target being free, the position tracking behavior will guide the robot again. These two behaviors are coordinated by a mediator (Dragoicea, 2003).

6. Concluding Remarks

Fuzzy logic methods have been proved to be effective tools to design highly responsive controllers for autonomous mobile agents. These controllers are capable of implementing motion and perception behaviors so as to attain multiple, possibly conflicting goals. Fuzzy - logic approaches have also been useful as the bases of formal results that facilitate the analysis of two important problems inherent in behavior-based approaches: the coordination of multiple behaviors (see (Dragoicea, 2003)) and the incorporation of prior knowledge into the controller.

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 vehicle intended for use in unstructured environments. The major problems that we have found with our approach concern the empirical nature of the definition and tuning of the fuzzy rules.

The focus of this work was on the application of fuzzy logic techniques to the design and implementation of basic navigation behaviors, and to the combination of basic behaviors to form complex behaviors to execute full navigational plans. For this, suitable simulation and real-time evolution environments are needed.

The MobLib open system provides a one-step solution in the sense that robot integration, controller design and numerical simulation are integrated into one platform for the ease of real-time implementation. This will greatly shorten the development life cycle and avoid the problem of compatibility and portability of codes. In addition, the software can also be used as a tool for teaching robotics courses. In this article, we present an overview of this open-architecture for mobile robot real-time control, discuss implementation issues, and provide a few case studies involving MobLib, the open - control architecture for mobile robots.

Results are available for the three cases above mentioned. These results were obtained on a real robot Khepera connected to the computer through a serial link. The fuzzy reactive multi-control strategy that is proposed in this paper allows obtaining very good results with less effort for developing fuzzy controllers for behavior implementation.

REFERENCES

1. BROOKS, R. B., 1986, **A Robust Layered Control System for a Mobile Robot**, IEEE Journal of Robotics and Automation, RA-2(1), pp. 14 – 23

2. DRAGOICEA, M., 2000, **Contribution to the Design of Adaptive Control Systems Using Neural Networks**, PhD Thesis, University Politehnica Bucharest

3. DRAGOICEA, M., DUMITRACHE, I., CUCULESCU, D.S., 2003, **Multi-behavioral Model Based Autonomous Navigation of the Mobile Robots**, International Journal Automation Austria, Vol. 11, Nr.1, pp:1-20, ISSN 1562-2703

4. DUMITRACHE, I., 1998, **Intelligent Control of Industrial Robots**, Ed. Mediamira

5. DUMITRACHE, I., 2000, **Intelligent Autonomous Systems**, Revue Roumaine des Sciences Techniques - Electrotechique et Energetique, vol. 45, No. 3, pp. 439-453, Bucarest

6. DUMITRACHE, I., DRAGOICEA, M., 2002, **Design of a Reactive Implementation for Autonomous Navigation of the Mobile Robots**, Conferinta Nationala de Robotica CNR'2002, 17-19 Octombrie, Craiova Ed. Universitaria, pp: 111 - 118

7. MURPHY, R. R., 2000, **Introduction to AI Robotics**, MIT Press