DEVS: AN ADD-ON FOR REACTIVE NAVIGATION

Youcef DAHMANI [(a)], Maamar El-Amine HAMRI [(b)]

[(a)] University Ibn Khaldoun B.P. 78, Zaaroura Tiaret, Algeria
[(b)] LSIS UMR CNRS 6168 University of Paul Cézanne Aix-Marseille III, France

[(a)] y@yahoo., [(b)] amine.hamri@lsis.org

ABSTRACT
This article discusses the use of the discrete event system specification (DEVS) to simulate reactive navigation. The article illustrates the utility of this formalism to combine behavioural robot navigation and systems modelling concepts.
In this work, we exploit the fuzzy logic theory in order to deal with imprecise and inaccurate robot localization. The data obtained from the localization module are presented to our DEVS model which is composed about three states representing respectively three behaviours: following left wall, following right wall and corridor following. Some modifications in Fuzzy Inference System are presented to optimize the calculus time.

Keywords: DEVS Formalism, Mobile Robots, Reactive Navigation, Fuzzy Logic Controller, Localization

1. INTRODUCTION

The mobility and the autonomy of robots pose complex problems, as regards generation of trajectory in strongly constrained and unstructured spaces. The other problem is of decision-making starting from information sensors vague or incomplete. To this end, robots need more sense, decision and technology (Michita 1999; Sergio 2000).
In this work, we use the DEVS formalism to describe three behaviours as three different states and the stimulus of each state is fired by localization distance which is given by fuzzy controller.
The DEVS (Discrete EVent system Specification) formalism was introduced by Zeigler (1976) as an abstract formalism for discrete-events modelling and simulation.
The DEVS formalism is a modelling approach based on systems theory. It's a modular and hierarchical formalism focused on state notion. DEVS is based on two types of models: atomic models and coupled models. Atomic model represents the basic behaviour of system and the coupled models are based on atomic models and/or coupled models, they represent the internal structure of the system which represent coupling between models (BISGAMBIGLIA 2008).
For the class of formalisms denoted as discrete-event (Fishwick 1995), system models are described as an abstraction level where the time base is continuous ($\Re$), but during time-span, only a finite number of relevant events occur. These events can cause the state of the system to change.

The Fuzzy logic permit to use mathematics concepts, its main advantage is the representation of the human been knowledge. The use of fuzzy logic gives good results in robot navigation without an analytical model of the environment.

2. THE DEVS FORMALISM

The DEVS (Discrete EVent Systems specifications) formalism allows two levels of description (Zeigler 2000; Glinsky 2004). At the lowest level, a basic component called atomic DEVS which describes the autonomous behaviour of a discrete-event system and at the highest level, a coupled DEVS which describes a system as coupled, hierarchical and modular model.

2.1. The atomic DEVS formalism

Formally, an atomic DEVS, which represents an atomic model, is specified by 7-tuple:
$AM=<X,S,Y,\delta_{int}, \delta_{ext},\lambda,ta>$
Where
X: input events set;
S: states set;
Y: output events set;
δ_{int}: $S\rightarrow S$: internal transition function, models the states changes caused by internal events, it describes the behaviour of a Finite State Automaton;
δ_{ext}: $Q\times S\rightarrow S$: external transition function, defines the state changes due to external events;
$Q=\{(s,e) \mid s\in S.0\leq e\leq ta(s)\}$: total states and ***e*** describes the elapsed time since the system made a transition to the current state s;
λ: $S\rightarrow Y$: output function, maps the internal state onto the output set;
ta: $S\rightarrow\Re$: time advance function, represents the lifetime of the state.

2.2. The coupled DEVS formalism

The coupled DEVS formalism describes a discrete event system in terms of a network of coupled components.
$CM=<\ ,D,\{\ \mid d\in D\},EIC,EOC,IC,select>$
Where

: set of possible inputs of the coupled model,
: set of possible outputs of the coupled model,
D : set of names associated to the model components,
| d∈D: set of the coupled model components, these components are either atomic or coupled DEVS model,
EIC: set of External Input Coupling,
EOC: set of External Output Coupling,
IC: defines the Internal Coupling,
Select: →D : function that defines priority between components.

3. FUZZY LOGIC CONTROLLER

A fuzzy logic controller permits to build control law from linguistic and qualitative description of system's behaviour via fuzzy base rules.
A fuzzy controller consists of 3 basic elements (Fig.1):
1. State interface (Fuzzification): numerical values are represented into linguistic variables with appropriate membership functions,
2. Action interface (Defuzzification): transforms the command actions into crisp values useable directly by the process which is modeled.
3. Inference engine: elaborates decisions from fired fuzzy rules, it's the core of the controller.

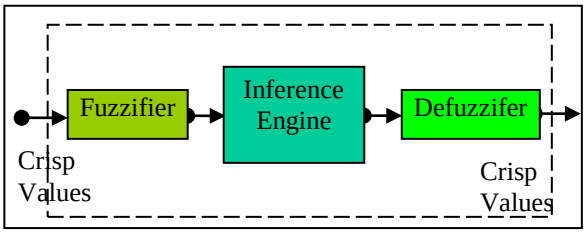

Figure 1: Fuzzy logic controller

4. MODULAR DESIGN

4.1. Robot Architecture

In the present work, the robot considered is circular having three sensors, one in front and one on each side. The sensor's orientation angle is 45° on both sides of frontal axis of the robot. For safety navigation manner, the robot is constrained by some points. The robot must move far from a safe distance, the corridor must be wide than a certain width, and the sensors have a limited scope (Fig.2)

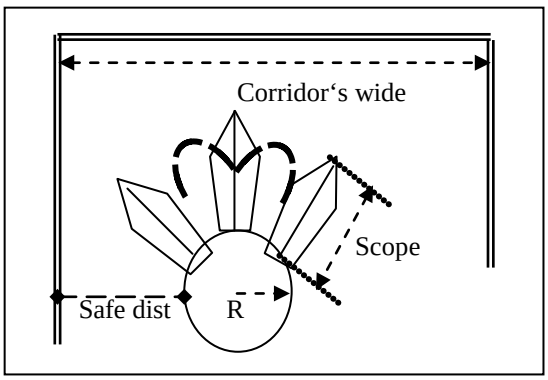

Figure 2: Structure and robot's sensor's position

4.2. Reactive Navigation in DEVS formalism

In this work, we have chosen 3 behaviours; each one represents a state (Fig.3). The transition from one state into the other is fired by external event (Table 1). This work is based on 6 events which are obtained from the localization module; this module activates the appropriate port to trigger the event. We denote Right Distance detection event (**rd**) gathered by the RD port, the Left Distance detection (**ld**) obtained via the LD port, Bilateral Distance detection (**bd**) which is gotten by by the BD port, Move to Target (**mt**) and the **end** event which are on respectively on the MT and the End ports.

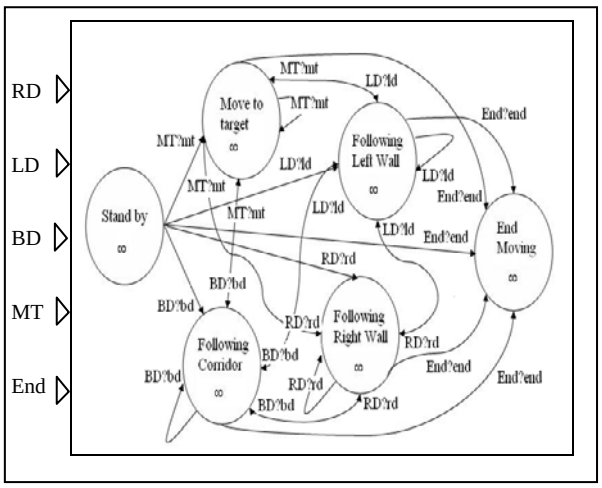

Figure 3: Reactive Navigation's DEVS model

Note that, the **bd** event is obtained if both events **rd** and **ld** are detected. **rd** and **ld** events are obtained by calculating the minimum distance between respectively, the right and frontal distance on the right sensor, and the left and frontal distance on the left gathered by the left sensor.
rd= min(right distance, frontal distance)
ld= min(left distance, frontal distance)
As illustrated on table 1, we define a following output function
δ_{ext}: Q×S→S
λ: S→Y

Table 1: Diagram of event's transitions

State / Event	Stand by	Tracking Right Wall	Tracking Left Wall	Tracking Corridor	Move to target	End Moving
rd	Tracking Right Wall	Tracking Right Wall	Tracking Right Wall	Tracking Right Wall	Tracking Right Wall	Not Available
ld	Tracking Left Wall	Tracking Left Wall	Tracking Left Wall	Tracking Left Wall	Tracking Left Wall	Not Available
bd	Tracking Corridor	Tracking Corridor	Tracking Corridor	Tracking Corridor	Tracking Corridor	Not Available
mt	Move to target	Move to target	Move to target	Move to target	Move to target	Not Available
end	End Moving	End Moving	End Moving	End Moving	End Moving	Not Available

4.3. Optimization of fuzzy controller process

The reproach of the fuzzy logic control usage is that it takes a lot of CPU time for calculations, especially for large number of fuzzy rules.

We introduce a discrete event version of Fuzzy Logic Control in order to reduce the fuzzy inference engine activity and speed up the calculation process. This idea is inspired from the works of Sheikh-Bahaei and Jamshidi (2004).

To do that, a change detector bit is added to fuzzy logic controller (Fig. 4). We define a change detector of each fired fuzzy rule, and if no change is observed on the participating linguistic terms, so we leave the rule and if rule or some other linguistic terms are hold we use the fuzzy inference method.

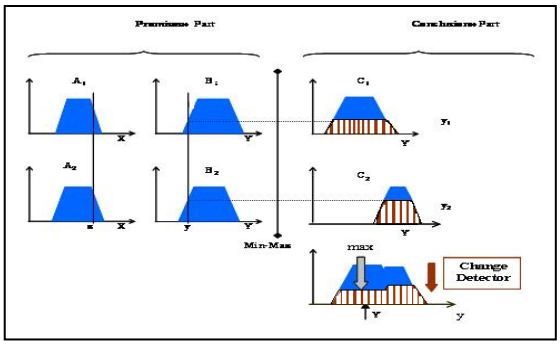

Figure 4: Proposed Fuzzy Controller

5. SIMULATIONS AND RESULTS

5.1. The Data Base Fuzzy Rules

According to our kinematics' model (Fig.5), we note θ the orientation of the robot, φ, represents the angle to the target, and α describes the deviation done by the robot when it moves (α=θ-φ).

So the structural fuzzy rules introduced by the Fuzzy logic controller are:

: **If** (θ-φ) is A **and** is B Then α is C

Where A,B and C are linguistic terms and Dx stand for Distance (right, left or frontal).

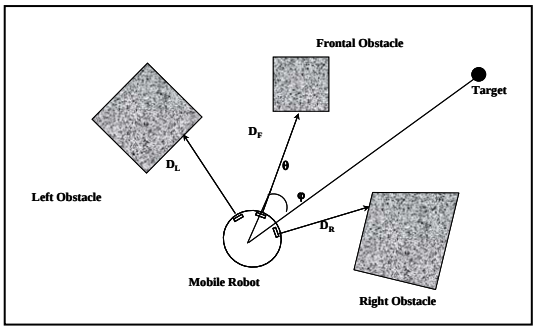

Figure 5: Cinematic of the Robot

The different variables are fuzzified as below (Fig. 6)

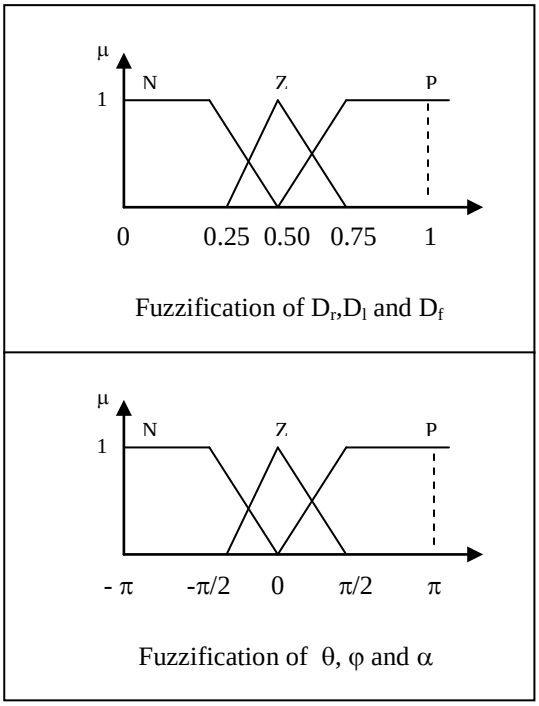

Figure 6: Input/Output Fuzzification

So, the data base fuzzy rules are described on tables 2a, 2b.

According to these rules, each behavior "state" is activated only when the change detector detects a change in the conclusion of the fired fuzzy rules.

Table 2.a: Fuzzy Data Base (Right Turn)

(θ-φ) / $\min_{(Dr,Df)}$	N	Z	P
N	P	P	P
Z	Z	Z	Z
P	N	N	N

Table 2.a: Fuzzy Data Base (Left Turn)

(θ-φ) / $\min_{(Dl,Df)}$	N	Z	P
N	N	N	N
Z	Z	Z	Z
P	P	P	P

5.2. Simulations and results

The figure 7 shows an example of a fuzzy rule implemented by these atomic models. Each atomic model has a change detector, such that each model is activated only when there is a change in the input, otherwise they are sleeping and don't take any CPU time.

We remark there is not a big difference between the two trajectories taken by the robot. But if we compare the activities between the 2 methods we found that we gain in number of iterative calculations evolving a diminution in process activity and in the other point, we decrease the number of event stimulus for the robot (see Table.3).

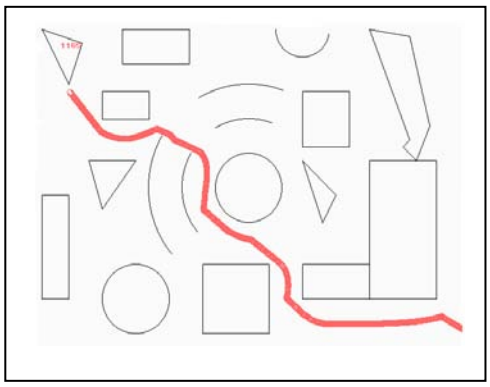

Figure 7.a: Conventional Fuzzy Navigation

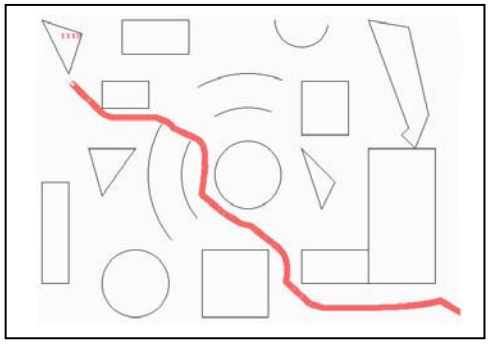

Figure 7.b: Optimized Fuzzy Navigation

The table below gives a comparison between the classic conventional fuzzy navigation and the combined discrete event one.

Table 3: Diagram of event's transitions

	Following right wall	
	Number of fuzzy iterations	RD Event call
Conventional fuzzy navigation	1765	49
Discrete Event	1119	26
	Following left wall	
	Number of fuzzy iterations	RD Event call
Conventional fuzzy navigation	814	9
Discrete Event	538	7

This result depicted on table 3 shows for the same environment, starting and ending at the same points (see Fig.7 a. and b.), the navigation process gives good results compared to the conventional fuzzy navigation.

We can see that we decrease the activity of cpu and calculation time by using the DEVS simulation. In conventional method all the fuzzy operations (fuzzification , inference, defuzzification) is done at every time step, but in Discrete Event Fuzzy Logic the calculation are done only if there is change (event) in the system.

6. CONCLUSIONS AND PERSPECTIVES

This paper has allowed us to combine fuzzy approach and DEVS formalism in reactive navigation; it gave us good results in particular for the 3 mentioned behaviours. However, the subject is not ready to be completed.

It's known that, we can always think of carrying out certain number of works, we can retain following work:

- ✓ Adding and simulating other behaviours with DEVS formalism,
- ✓ Realization of conflict behaviour in robot navigation,
- ✓ Integration of the vague concepts on all levels of robot's architecture,
- ✓ Trying this method in more complex environment.

REFERENCES

Bisgambiglia, P.A., 2008. *Approximate Modelling Approach for Discrete Events Systems: Application for Forest Fire Propagation*, PhD Thesis, University of Corse, France.

Fishwick, P., 1995. Simulation Model Design and Execution: Building Digital Worlds.

Glinsky, E. and Gabriel, W., 2004. Modeling and Simulation of Hardware/Software systems With CD++. *Information Processing and Management*, 7 (2), 147–168.

Michita, I., Kazno, H. and Tsutomu, M., 1999. Physical Constraints on Human Robot Interaction. *Proceedings of the International Joint Conference on Artificial Intelligence*, July-August, Sweden.

Sergio, U.G. and Horacio, M.A., 2000. Application of Behavior-Based Architecture for Mobile Robots Design. *Lecture Notes in Artificial Intelligence 1793, MICAI 2000: Advances in Artificial Intelligence*, pp 136-147, April 2000, Mexico.

Sheikh-Bahaei, S. and Jamshidi, M., 2004. Discrete Event Fuzzy Logic Control with Application to Sensor-Based Intelligent Mobile Robot. *Proceedings WAC*, June 28-July 1, Spain.

Zeigler, B.P., 1976. *Theory of Modeling and Simulation*, Academic Press.

Zeigler, B.P., Praehofer, H. And Kim, T.G., 2000. *Theory of Modeling and Simulation Second Edition Integrating Discrete Event and Continuous Complex Dynamic Systems*, Academic Press.

AUTHORS BIOGRAPHY

Youcef DAHMANI is lecturer of computer science at the Ibn Khaldoun University of Tiaret. He obtained his diploma of computer engineering in 1992 from U.S.T.Oran, Algeria, and the MSc degree in 1977 from university of Es Senia Oran and received the doctorate in 2006 from U.S.T.Oran. His research areas include optimization of fuzzy rules, artifial intelligence, reactive robotic systems and network security.

Maamar El-Amine HAMRI is an associate professor at Aix Marseille III university and a member of LSIS lab. His main research is the discrete event simulation. Currently his is interested to the use of simulation in IA and software engineering. He is also member of the M&S network and supervises the M&S dictionary project.