# Reactive Robot Development: An Exploratory Approach

João Pedro Dias
INESC TEC
FEUP Campus, Porto, Portugal

Email: jpmdias@fe.up.pt

***Abstract*—Reactive robotics is a subfield of robotic study field that tries to control robots by the description of behaviors (inspired on humans). There are several architectures for implementation such reactive robots, and, in this paper, the Brook's subsumption architecture is followed. By the end, it is shown that we can easily use one robot simulator to explore this architecture, with an application scenario where a robot should follow a wall in D-format.**

## I. Introduction

Robotics is defined as the study of intelligent connection between sensing and actuation, taking into account considerations about modeling, planning, knowledge, reasoning, and memorization, resulting in a *thing* with intelligent properties [1].

There are mainly three robotic development paradigm, namely, hierarchical/deliberative paradigm, reactive paradigm and hybrid deliberate/reactive paradigm.

First, in the hierarchical/deliberative paradigm [2], robots follows a top-down fashion, consisting heavy on planning. As so, the robot senses its surroundings, plans the next action and acts on the world, planning explicitly the next move and gathering the data tends into a global world model.

The reactive paradigm [3], also known as behavior-based robotics [4], consists of robots that are instructed to perform actions by the use of low-level primitive behaviors. Complex behavior then arises from the interaction of the behavioral set and the complexities of the environment in which the robot finds itself, providing more rapid and flexible response than the possible by traditional methods of robotic control [5].

Lastly, in the hybrid deliberate/reactive paradigm [6], the robot first plans (deliberates) what is the best way to decompose a task in subtasks and then selects what are the most suitable behaviors to accomplish each subtask. Behaviors are then executed following the reactive paradigm. In this paradigm, the sensing process follows a mixture of hierarchical and reactive paradigms, being the sensor data router to each behavior that needs that sensor, but being also available to the planner for construction of a task-oriented global world model.

Reactive robots were originally pursued as a way of modeling human problem-solving methodologies. Although, reactive robots have been successfully implemented by the use of rule-based systems (i.e. control strategies in which the robot exhibits behavior as reaction to events in the environment[7]). Such approach is used, for example, in the programming of the NASA's Mars rover [8].

There exists a set of alternative architectures for defining and programming reactive robots being the best known the brooks's subsumption architectures [3], popular in the 1980s and 90s. This architecture proposes a division of rules into various levels of competence. An example of such architecture can be described as follows:

- *Bottom level*: Ability to make a robot move randomly;
- *Middle level*: Ability to avoid obstacles;
- *High level*: Ability to define goals (e.g. move towards a particular position).

On every execution cycle, rules are enabled for firing based on the current robot state and sensor readings. Additionally, the rule's level of competence determines its priority, being always the highest-level enabled rule triggered.

Another architecture is the *Timed FSM* (Timed Final State Machine) architecture [9]. This architecture consists on using final state machines alongside with timeout restrictions. A state machine consists on a reactive system in which it response to a particular stimulus (e.g. sensor reading) is not the same for every situation, depending always on its current state. The existence of time-out mechanisms guarantees that the state machine moves from state to state when needed, even when an external *stimulus* is not present. An example scenario can be when a robot is on it `move` state it and a sensor gives an alert of a wall being close to it, it transits to the `stop` state.

Another important topic worth mentioning is the importance of simulation in robotics, namely, as way of easily design, develop and testing robots. There are a lot of well-known robotics simulators ranging from open-source to commercial ones. To name a few open-source ones [10]: Player/Stage, Gazebo, ROS, Simbad, CARMEN, USARSim, MRDS, MissionLab. Such simulators typically have a set of features, namely:

- Fast robot prototyping (e.g. using Visual Programming Languages [11]);
- Physics engines for realistic movements (e.g. gravity);
- World rendering (3D or 2D depending on the use case);
- Dynamic robot bodies with scripting [10].

The current paper describes the exploratory work done about the reactive robot development. The main objective is

to develop and simulate (using any of the available robotics system simulation available) a simple and 2D robot that can follow a wall in a D format from inside and outside. The approach used follows the Brook's subsumption architecture.

The structure of this paper goes as it follows, firstly it is given an overview on the architecture of our approach as well as the technologies used in Section II. Then, the results obtained are presented in Section III and a critical overview on limitations of our approach in Section IV. At last, some closing remarks are presented in Section V.

## II. Architecture Details and Implementation

The scope of this paper can be summed as:

> A *simple* reactive robot, running in a 2D simulated environment, that is capable of wandering until it finds a wall and then follow the wall forever within a D-formated world map. It must work inside the D wall and also on the outside of it.

For implementation such robot, firstly there was the need of picking a technology to develop it. There was a few additional requirements that we defined, as follows:

- Open-source solution;
- Low-code;
- Simple simulation engine that can be easily understandable;
- Focused on 2D world simulations.

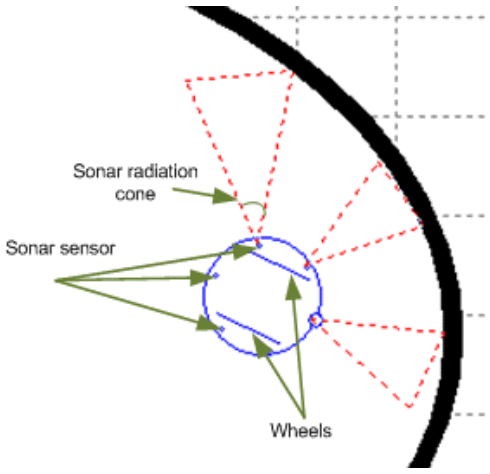


Fig. 1. An example of a robot configuration in Azolla simulator [12].

This additional requirements made us pick the *Azolla*, a 2D LUA-based mobile robot simulator implemented in C++. This simulator allows one to design how a robot will navigate in a 2D world by using a set of rules defined by us [12]. A simple presentation of the robot schematics (sensors and motors) is given on figure 1.

The rules are designed by using Lua programming language, that is a powerful, fast, lightweight, cross-platform and multi-paradigm embeddable scripting language [13].

At the designed architecture, since it follows a subsumption architecture, it was defined a number of rules that the robot obeys under following a set of apriori defined rules. The rules can be enumerated as follows, being the last the ones with higher priority:

1) Robot wanders in the world;
2) If the robot meets a wall, the robot rotates until it is in parallel position with it;
3) The robot should always stay close to the wall within a apriori defined interval.
4) If the robots goes to close to the wall, it should step away from it.

Following this designed rule system in the most simple way possible, a simple Lua script was developed mapping this rules, using a set of *if*, *elseif* and *else* clauses.

## III. Results

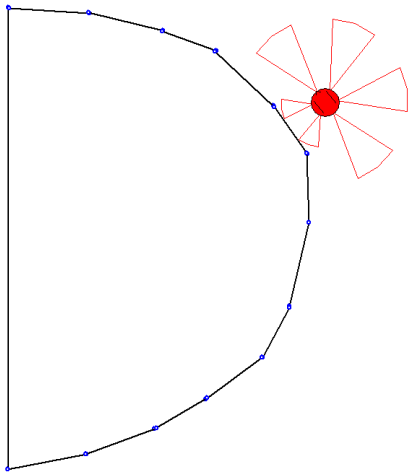

Fig. 2. Robot following the D wall from the outside.

The resulting robot can be observed in Figures 2 and 3 interaction with the D wall world. It constantly follows the wall after it found it was defined in the rules of the previous chapter. The robot keeps readjusting its movement to fit the wall curves and particularities.

## IV. Limitations

Even with the meet of the initial objectives of doing a simple robot that follows a wall in a D format, a lot of improvement could be made. The limitations can be divided between the limitations of our approach and limitations of the simulator itself.

At our developed approach we can identify three principal limitations:

- The robot speed does not adjust to the current path and is always constant.
- The robot have some problems wandering in the world, making loops and not finding the wall correctly.
- Some exceptions can occur when the robot is positioned in space-constraint situations (e.g. corners).

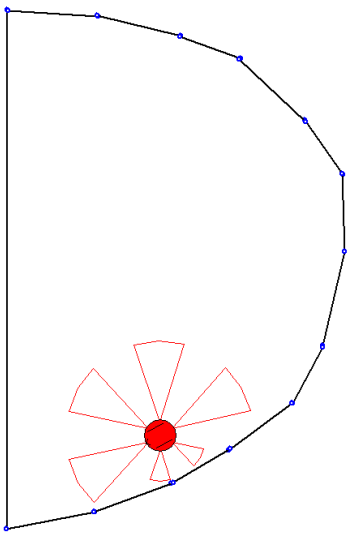

Fig. 3. Robot following the D wall from the inside.

At the simulator level we can identify several limitations, that can be corrected, since this is an *open-source* project:

- The world designer is hard to use, only allowing the drawing of straight lines, rectangles and circles.
- The simulator is not a real time robot simulator and so, if we add more and more robots, the simulation will run slower.
- The sensor reading values is based on pixel reading of the screen and if the robot goes out of the main window, the sensor algorithm will read the wrong screen pixels.

## V. Conclusion

Robotics is a field that has everyday more impact on our everyday lives. Different robots are performing repetitive tasks, and, even further, doing more advanced tasks with the use of Artificial Intelligence mechanisms.

Reactive robotics is a valid approach when developing robots that should follow a set of behaviors depending on the context and tasks they have to perform.

Our exploratory work shows how a subsumption architecture can be followed to develop a robot that follows a set of rules with different priorities in order to follow a wall.

The resulting implementation have limitations. This limitations are result of deficiencies on the rule system, but, as well, from some limitations of the picked simulator.

Further work must be pursued at the subsumption rule system developed, but also, contributions to the *open-source* simulator can be done in order to improve it, while maintaining its simplicity.

## References


[1] Bruno Siciliano and Oussama Khatib. *Springer handbook of robotics*. Springer, 2016.

[2] Robin Murphy. *Introduction to AI robotics*. MIT press, 2000.

[3] R. Brooks. A robust layered control system for a mobile robot. *IEEE Journal on Robotics and Automation*, 2(1):14–23, Mar 1986.

[4] Ronald C Arkin. *Behavior-based robotics*. MIT press, 1998.

[5] Ronald C Arkin. Reactive robotic systems. 1995.

[6] Yingzi Zeng, Yanan Li, Pengxuan Xu, and Shuzhi Sam Ge. *Human-Robot Handshaking: A Hybrid Deliberate/Reactive Model*, pages 258–267. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[7] Arcot Sowmya. Real-time reactive model for mobile robot architecture. In *Aerospace Sensing*, pages 713–721. International Society for Optics and Photonics, 1992.

[8] S. Y. Harmon. A rule-based command language for a semi-autonomous mars rover, 1990.

[9] Marco Della Vedova. Robotics - finite state machines, 2015.

[10] Marc Freese, Surya Singh, Fumio Ozaki, and Nobuto Matsuhira. Virtual robot experimentation platform v-rep: A versatile 3d robot simulator. *Simulation, modeling, and programming for autonomous robots*, pages 51–62, 2010.

[11] J. Jackson. Microsoft robotics studio: A technical introduction. *IEEE Robotics Automation Magazine*, 14(4):82–87, Dec 2007.

[12] Auralius Manurung. Azolla - 2d lua based robot simulator, 2014.

[13] Roberto Ierusalimschy. *Programming in lua*. 2006.