Robotics
2015
Practical Challenge 2
Subsumption architecture, Logic-labelled finite-state machines, and proportional derivative control.

Vladimir Estivill-Castro
MiPal

October 21, 2015

Objectives

1. Being able to program a robot to carry out some simple *PD* control.
2. Practice more Logic-labeled finite-state machines (*llfsms*) to construct a behavior.
3. Practice more with robotic middleware and the PUSH-approach of `ROS`.
4. Introduction to the subsumption architecture.

Warning

Be aware that every single piece of software used here is software in evolution, like all software. Note that MacOS rapidly has migrated rapidly (from Leopard, to Mountain-lion to Mavericks, to Yosemite to El-Capitan), Ubuntu has moved in a few years several versions. Laptops have moved from 32-bits to 64-bits. `ROS` has moved from Hydro to Indigo to Jade. `Webots` from 6.X to 7.X, to 8.x. Thus, you have to be patient and resilient. The *MiPal* software aims for being as cross-platform and as standard as possible. This is perhaps the very first lessons of robotic software architectural styles, there are many software components of not necessarily compatible versions.

This challenge should not require the *MiPal* compiling infrastructure. The aim is that it should be completed with the released versions for `ROS`, and every other shareware.

Background

You should have some experience or you may want to refresh on the following.

1. Laboratory One.
2. POSIX tools and UNIX shell (specially Ubuntu and the MacOS terminal).
3. `C++11` programming. You may need background in threads and tools that avoid busy waiting of the CPU.

Instructions

This laboratory sessions has three (3) parts. It assumes you have completed the previous lab.

1. First task is a reactive control using the light sensor.
2. Second task is a proportional derivative control using the sonar sensor.
3. Third task is a hierarchy of *llfsms* where one machine controls which of two other sub-behaviors is running..

This laboratory exercise requires you to investigate several topics. You are invited to place references to the literature in your report. Most importantly, you are to provide feedback on the instructions and the documentation of *MiPal*'s tools (feedback on this text is also welcome).

Tools

1. You should have all the tools you used in Laboratory One.

First task.- Reactive Control — Light sensor

The idea of reactive control is to respond to a setting of sensor with the most appropriate action. No knowledge representation, and no reasoning (no deliberation).

Your job is to develop a **reactive-control** software-module that creates such control without any deliberative control. The desired behavior of the robot is very similar as the behaviors you developed in Laboratory 1.

1. If no signal is received, the robot moves forward.
2. If the button attached to the right-bumper is pressed, then the robot backs-off with both wheels turning backwards, makes a small turn to the left. Then, it continues with the default behavior going forwards.
3. If the button attached to the left-bumper is pressed, then the robot backs-off with both wheels turning backwards, makes a small turn to the right. Then, it continues with the default behavior going forwards.
4. If the sonar sensor detects an obstacle, it backs-off, randomly chooses a direction to spin and then makes a slight turn in such random direction before going to the default behavior of going forwards.

5. If it detects a change of light in the surface it is running, it backs of very little, makes a complete spin of 180 degrees, and continues running forward.

The controller should be completely structured as a **reactive-control** architecture.

Second task.- Proportional Derivative Control — Sonar sensor

You are to construct a behavior so that the robot maintains a distance of 30cm from an obstacle once you push one of the bumper sensors. If the obstacle advances to the robot, the robot backs maintaining the distance of 30cm. If the obstacle moves away from the robot, the robots should seem to chase the obstacle.

The requirements are that you should create the behavior as a logic-labeled finite state machine. You should create several version of the behavior. You may need to extend the `NXT_dirver` to create these variants.

Proportional Control: The *llfsm* controlling the robot issues commands that turn on the motors but indicate a **distance** that is to be covered by the robot. For example, if the distance in the sensor is 32cm. The error would be 2cm, and perhaps the gain is 0.5. So the commands the *llfsm* sends to the robot is to *move one cm forward*.

Proportional Derivative Control: The *llfsms* controlling the robot issues commands that turn the motors but indicate **the power** to the motor proportional to the error distance. So, if for example, the sonar reports 32cm and the gain is 0.5, then the *llfsm* instructs the motors to be at 1% of their power forwards.

You must record videos with different gains that show the behavior of the robot sometimes to slow to react (small gains) or actually to eager to cover the distance (large gains).

Third task.- Hierarchy of *llfsms*

The compiler `clfsm` can run *llfsms* in a fixed sequential schedule but concurrently, and arrangement of *llfsms*. In the arrangement, one machine can `resume`, `restart` or `suspend` another machine. Create a master machine that starts the behavior of the first task (using the light sensor) if the left-bumper button is pushed and performs this behavior until this bumper is pressed again. However, it performs the behavior of the proportional derivative control if the right bumper button is pushed (and until this right button is pushed again).

> **CAUTION:** The master behavior and the behavior of the first task have somewhat contradictory reaction to a button of the bumper being pressed. See what can you imagine as creative solutions for this.

What shall you submit or demonstrate

You should submit a *Practical Challenge Report*. You should demonstrate you completed all the activities in the requested tasks. It is OK to complete the labs with your own robots if you are so inclined and acquire `LEGO-NXT` robots.

You should submit a reflective report on the experience in the lab with enough evidence that shows you have completed all the activities. You may include some screen shots. You may include links to videos. **The challenge report must be a PDF document.** Please discuss in your own words the topics covered in the challenge. That is, describe concept like

1. arrangements of *llfsms* and subsumption architecture,
2. finite-state machines, and/or
3. closed-loop control.

It is important to describe how would you have implemented the behavior of the last tasks (the NXT robot) if you didn't have access to `clfsm` and to a module like the `NXT_controller`. That is, what would be the structure of a program that achieves that behavior if it had evolved from the `testDemo` program of the `NXT_dirver`? What would be the software architecture in such case?

Some other important concepts you should investigate further are as follows.

1. The relevance of time in relation to the software controlling a robot.
2. The basics of feedback-loop control.
3. The basic of the subsumption architecture.