Technical University of Crete
School of Electrical and Computer Engineering

INF 412 – Autonomous Agents – 2023

Instructor: M. G. Lagoudakis **1st Laboratory Exercise** Deadline: 13.11.2023, 23:59

Introduction

A large class of robotics software is that of robot simulators. These applications simulate the kinematic and dynamic state of various robotic systems within virtual worlds and enable users to test control algorithms and experiment with various techniques, avoiding the costs and risks involved in experimenting with the corresponding physical robotic systems. This way, the development of robotic software becomes easier and the code debugging process is sped up. Additionally, simulators play an important role in the employment of robotic learning algorithms, as they can relatively easily provide the large amounts of data required by these algorithms, without straining real robotic systems and without a huge investment of valuable time. However, criticisms have been raised from time to time that usually relate to the extent to which a simulator succeeds in realistically simulating every aspect of the real world, including the inherent uncertainty of the physical world. Sometimes small deviations in the simulator can lead to huge deviations of results in the physical system. Despite the problems, simulators are and will continue to be an important part of robotics research. The goal of this lab is to familiarize you with one of the most-established simulators, the Webots simulator, which was originally developed at EPFL in Switzerland and is now commercially available through the company Cyberbotics [www.cyberbotics.com] also based in Switzerland.

Installation

To work with the Webots simulator, you will need to download:

- the installation file for the latest version **2023b** from www.cyberbotics.com
  (installation files are available for Windows, Linux, Mac)

Webots

The Webots Simulator provides an integrated development environment for robotics applications. In fact, the user has four basic development stages available: Model, Program, Simulate, Transfer. In the modeling stage, the user can create an accurate model of a robotic system with all its physical characteristics (joints, parts, sensors, actuators, size, weight, friction, etc.) or choose to use one of the already defined models for a multitude of commercial robotic systems, such as Aibo, Nao, or Atlas. Then, through an integrated graphical software development environment, the user can program the robotic system using one of the many available programming languages and the corresponding built-in tools for compilation and debugging. The execution of the code can then be tested through the robot model in any virtual world to assess the final behavior of the robotic system under various conditions. Finally, this code can be relatively easily transferred to the real robotic system using the corresponding tools for compilation and optimization. The Webots simulator offers many facilities to synthesize robotic parts from libraries of sensors and actuators, realistic simulation of kinematic and dynamic states according to the laws of solid state physics, acceleration or deceleration of simulation time, and interactive 3D visualization and communication with the user. More information is available at www.cyberbotics.com.

Manuals

The Webots manuals (also accessible from the Menu `Help`) include instructions for learning how to use it, as well as technical details for each supported robotic system.

Webots User Guide - Simulator user guide www.cyberbotics.com/doc/guide/index

Webots Reference Manual - Reference Manual www.cyberbotics.com/doc/reference/index

Procedure

After installation, launch the Webots simulator on your computer. Take the time to complete the Webots Guided Tour (if it doesn't start automatically, start it from the Menu `Help`) to see what it has to offer (keep the tour window open and use Previous/Next to view at least the 19 robot demos, while devices demos can be viewed selectively). Read the information given for each environment and each robotic system. Use the mouse to change the viewpoint in the 3D world and the control keys (where offered—look for instructions in the console window) to guide the simulated robot. On the right you can see the code being executed and at the bottom messages displayed during execution. On the left you see all the objects that exist in the current 3D world, as well as their properties. Read in the User Guide about the functions of the User Interface so that you can operate it easily.

RobotStadium
The RobotStadium (robotstadium.org) is a robotic competition that was officially held in the years 2008–2011 and is essentially online simulated RoboCup Standard Platform League (SPL) matches with four humanoid Aldebaran Nao robots per team. It is an exciting competition that allows studying all sub-problems of RoboCup (perception, localization, motion, coordination) in a fairly realistic environment. Participants can program their team in different languages (C++, Java, Python, URBI). In subsequent years, the ability to program and simulate humanoid robots Robotis Darwin-OP has been added, and of course the ability to hold matches between teams of Nao and Darwin-OP robots.

Experiment
Copy the folder `webots/projects/robots/softbank/nao` to some user space on your disk without Greek characters in the path (all references hereafter will be to your copy). From `File` choose to open the world `robotstadium_nao_vs_robotis-op2.wbt` located in the `nao/worlds` folder. You will see in the GUI the field with the 5 Nao robots (four players and a goalkeeper) on one team and the 5 OP-2 robots on the other team. The red group of Naos is controlled by the `nao_demo` controller, whose code is written in C and is located in the `nao/controllers/nao_demo` folder, while the blue group of OP-2s is controlled by the `void` controller, which does nothing. More specifically, the `nao_demo` controller allows you to control the robots using the keyboard (see the related console instructions). By selecting a robot and opening its properties from the scene tree on the left, you can see which controller is used, and `Edit` opens its code window on the right. Try out the available features and see how they are implemented in code. Press `Run` to run the game simulation (the simulation cycles through the Initial, Ready, Set, Play states before the game begins). Control a player and try to score. Try making changes to the `nao_demo` controller code and then press `Clean` and `Build` to build the new executables and restart the simulation with `Reset`.

Exercises
Now it's your turn to program a simple behavior in Webots for the Nao model. The ready controller does not control the robot autonomously, but waits for your commands from the keyboard. Can you build your own controller that uses information from the sensors and guides the robot autonomously? To create your own controller, you can use the `nao_team_1` controller as a base, which is written in Java. It can be downloaded from eClass and should be placed in the appropriate subfolder in your working folder. See also the related instructions in the User Guide for installing the Java JDK and setting the paths in the environment variables on your machine. Before trying this controller, it is necessary to make two changes to the world `robotstadium_nao_vs_robotis-op2.wbt`: in the scene tree on the left, change the elements `TexturedBackground` and `TexturedBackgroundLight` from `stadium` to `mountains`! (Can you find the reason for that?) You can also delete the players you don't need to reduce the computational load. After making these changes, save the new world with a name of your choice to keep it.

The `nao_team_1` controller gives a good and fully autonomous soccer player behavior. It contains functions for visual recognition of the ball, for movement towards the ball, for alignment with the goal, etc. During the simulation, a double click on a robot opens the robot window, where you can see almost all sensor readings of the respective robot. Now, use your imagination for the desired behavior of your player. See if you can achieve what you want with modifications to the code and other fixes as needed. Try to play somewhat aggressively, avoid the player in front of you, kick the ball, get up if you fall, and, why not, score a goal! Okay, we're exaggerating... Of course, no one expects you to create the perfect soccer player, but give your robots some autonomy, so they don't wait for orders from you, and help them play elementary soccer. If nothing else, you must at least incorporate the shoot motion, which this controller doesn't use at all.

Report/Delivery/Rating
Compress your working folder `nao` which also includes your code (controller). Record a video of your player's final behavior through Webots (see `File`), but make sure the video file size is kept small, say under 10MBs. Write a short report (in PDF) describing what you added or modified. Finally, submit code, report and video as three separate files via eClass in the Section Assignments. Your grade will be determined by the completeness of your work ... and the originality of your player!