Running the Robotito using SCS Platform

David Ehrenhaft

University of South Florida

Introduction

This guide explains how to run the Spatial Cognition Simulator (SCS) models using the Robotito Robot.

It is assumed that you have read and completed all steps in the installation guide before coming to this guide. If not, confer with that guide before continuing.

Note: for all terminal commands containing the keyword “ros”, you must first enter “source /opt/ros/kinetic/setup.bash” in every terminal using such commands, unless that line is added to your bash file.

Preparing the physical setup

1. Replace the battery in the Robotito
    a. The batteries used are 2 cell LiPo batteries (voltage ~5V). Any such battery will work, regardless of the amperage.
    b. Unscrew the bottom platform and place the battery inside the chamber on the Velcro patches. Replace the bottom platform.
    c. When you connect the battery to the robot, an LED on the teensy board should flash and two different pitches of beeps should sound. This signals that the robot is receiving power.
    d. Confirm that the Robotito’s current firmware is the firmware file found in “~/catkin_ws/src/robotito/firmware/”. If unsure, upload this file.
    e. To save battery power, the Robotito should be unpowered until you are ready to use it. Simply disconnect the battery at any time, as the Robotito does not save any data used in the SCS setup.

2. Set up the maze and markers
    a. Layout a maze that you wish the robot to complete – this can be a simple pool with outer walls, or can include any number of corridors. For the Multiscale Actor Critic Morris Maze Model (the model this extension was designed for), you should simply create a complete perimeter for the arena, with no internal walls.

b. The setup currently uses a number of Alvar AR markers to track the robot's position and orientation and the positions and orientations of the walls. Many of these have already been printed and placed throughout the lab. If any have been lost, you can print more using the image files found in ~/catkin_ws/src/robotito/tags/.

c. Marker 0 should be on a 3D-printed platform, which can be placed on top of the Robotito. This marker should be placed such that the white line through the black square faces the Robotito's left side.

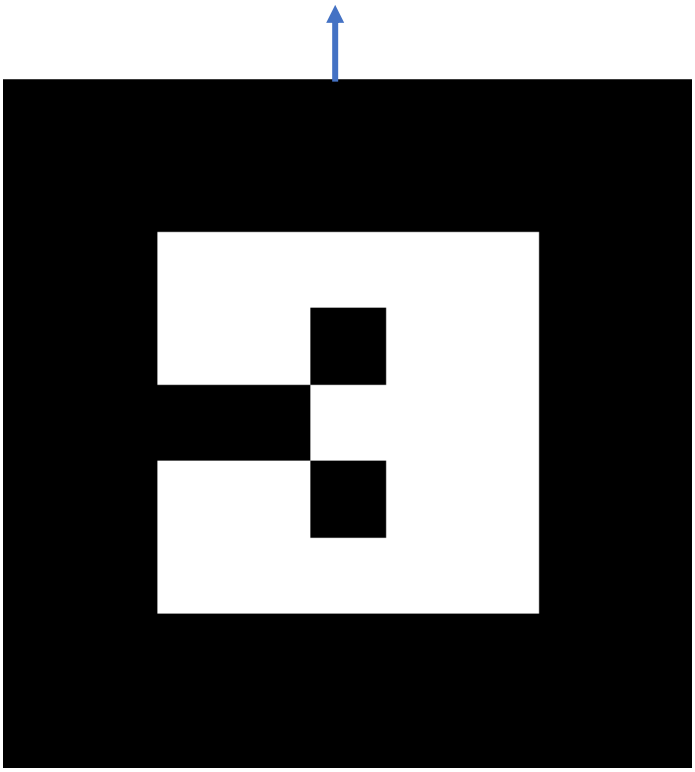

Figure 1: AR marker 0; the blue arrow should face the robot's forward.

Note: On the older Robotito model used, the robot's "back" is the side where the power cords are. If unsure, send a test packet using XTCU with the contents "v 90 80 80." This packet moves the Robotito directly forward.

d. Markers 1, 2, 3, and 4 are placed on cardboard platforms. These are currently used to mark to outer boundary walls. Place these on the outside of the walls, such that the characteristic white lines in the square face towards the center.

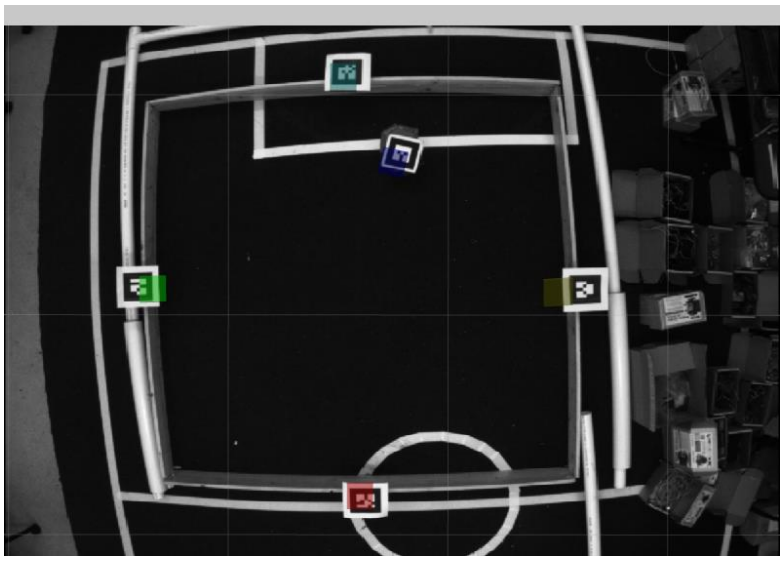

Figure 2: Overhead camera picture of simple maze setup

e. Markers 5-10 are currently printed and cut in the lab and can be used to mark extra external or internal walls and simulate obstacles (see ROSWallDetector.java in SCS to change what these values mean). These should be placed such that the center of the marker covers one end near the center of each extra wall.

f. Refer to scs/multiscalemodel/src/edu/usf/ratsim/robot/robotito/ROSWallDetector.java to modify the walls sizes, obstacle sizes, obstacle widths, and what type of entity each specific value represents as necessary.

3. Plug in the first Xbee using a USB cord and an Xbee platform to the machine running SCS.
   a. You will probably need to update the port that SCS uses in Robotito.java.

Note: if you need to restore the profile for the Xbees, refer to the Installation Guide, page 5, step (5). This explains the process in detail.

Running the setup

1. Ensure the XML file you will be using has the proper parameters. The XML file used for the Multiscale Actor Critic Morris Maze Model is found at ~/scs/multiscalemodel/src/edu/usf/ratsim/model/morris/multiscale/Robotito_Experiment.xml
   a. The robot must be set to `edu.usf.ratsim.robot.robotito.Robotito`
   b. The universe must be set to `edu.usf.ratsim.robot.robotito.ROSUniverse`
   c. You must set a start position using the `edu.usf.experiment.task.robot.PlaceRobotInitally` task for each group.

d. You must set the platform position using the `edu.usf.experiment.task.platform.CreatePlatformROS` task for each group.

e. You must set the size of the maze (it's at the end of the "model" tag)

f. To determine the coordinate for c and d, you will need to use "rosotopic echo /Robotito/pose" and use the robot's marker to determine the value for the desired position in the physical maze.

Figure 3: XML File Snippets for important setup parameters

2. Open a new terminal on the CMAC1 and run roscore:

   a. "roscore"

3. Open another new terminal and launch the ROS node setup:

   a. "source catkin_ws/devel/setup.bash", if you haven't modified your bash file to do so automatically

   b. "roslaunch robotito pose_detector.launch"

> Note: To confirm that these two ROS modules are running, try "rosrun rviz rviz" in a new terminal. Once RViz opens, click add in the lower left corner. Go to the 'by topic' tab and add the camera and the marker visualization. This should give a feed from the camera and highlight the AR markers found. Figure 2 is an example of the output you should see.

4. Run the SCS software using Eclipse (refer to this guide to set up Eclipse for SCS) or using terminal set the the SCS directory by typing the following:

"./scripts/visualExec
multiscalemodel/src/edu/usf/ratsim/model/morris/multiscale/Robotito_Experiment.xml
logs/Physical/Learning1/ Learning 0"

Note: In the above command, the second argument is the XML file used, the third is the folder to log results to, the fourth is the type of group (in this model, either Learning or Control), and the 0 is the number assigned to the individual. You will probably use the first argument for all experiments, given the length of each individual.

5. Once SCS is running, a java applet window should appear. The robot symbol (A circle with a line on one end) should not be moving or inside any black lines (as depicted in Figure 4). If the robot is moving, or walls are already drawn, confirm you are using the Multiscale Actor Critic Morris Maze Model and that you are using the XML file listed in the previous step.
    a. The console will direct you to move the robot to a specific position and press enter. To determine the Robotito's (marker's) current position, open a new terminal and enter:
        i. "rostopic echo /Robotito/pose"

    which will print the robot position and orientation in a stream.

6. You will be prompted to turn the robot to the specified orientation, then press enter.

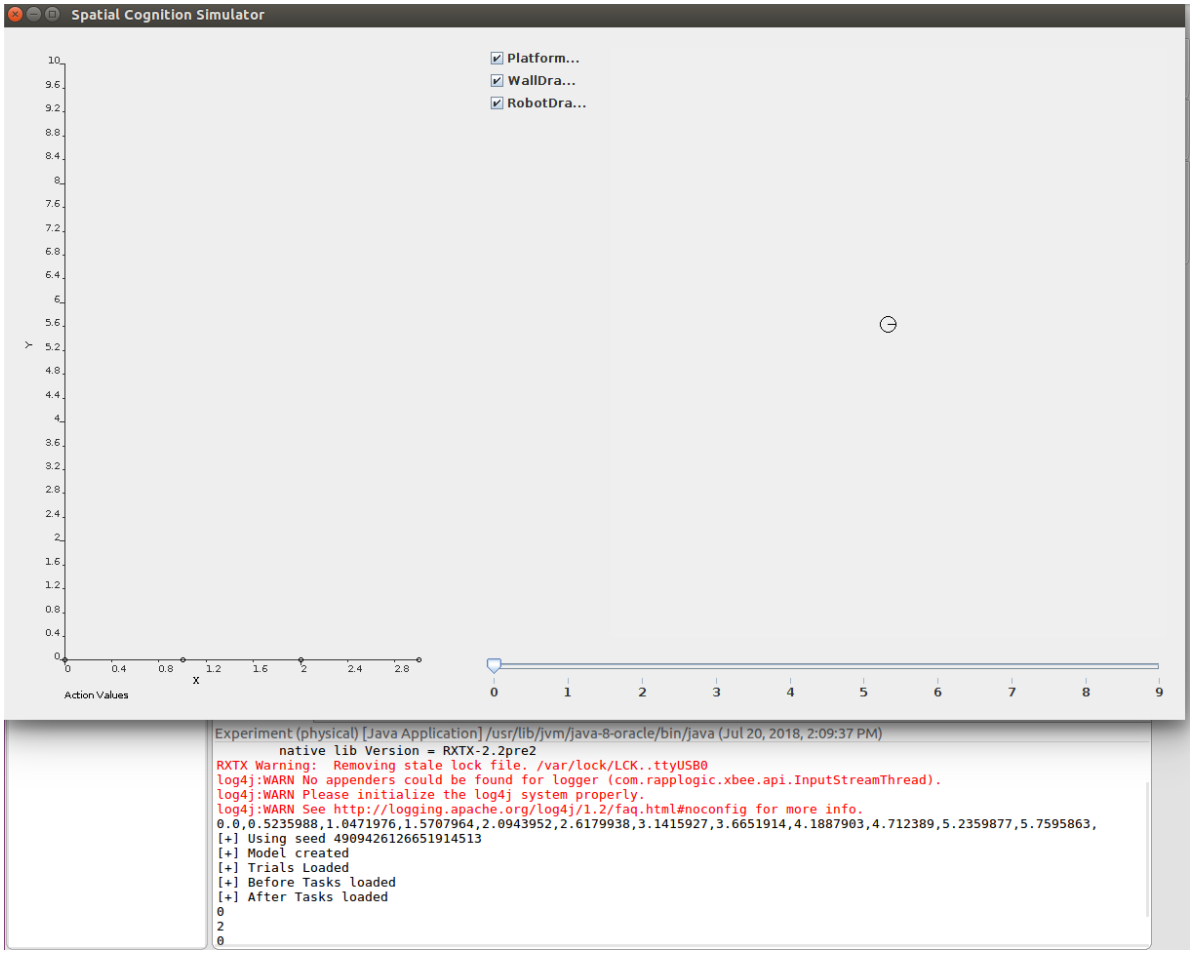

Figure 4: The SCS application window before initial robot placement and the initial console information.

7. The simulation applet will draw the detected walls and the robot will begin running episodes (refer to Figure 5). At the conclusion of each episode, the robot will stop moving and the console will again prompt you to move the robot to the initial position and orientation. This will continue until the number of episodes and individuals reaches the number specified in the experimental XML.

    a. Between the time the Robotito reaches the goal and when you press enter twice, SCS is stalled. You can use this time to check the battery voltage or to replace batteries, or to realign or add markers.

Note: You will probably see warnings regarding the loggers, an RXTX version mismatch, and possibly RXTX warnings about removing a stale lock. These are all harmless to the execution of the program, so you may safely disregard them.

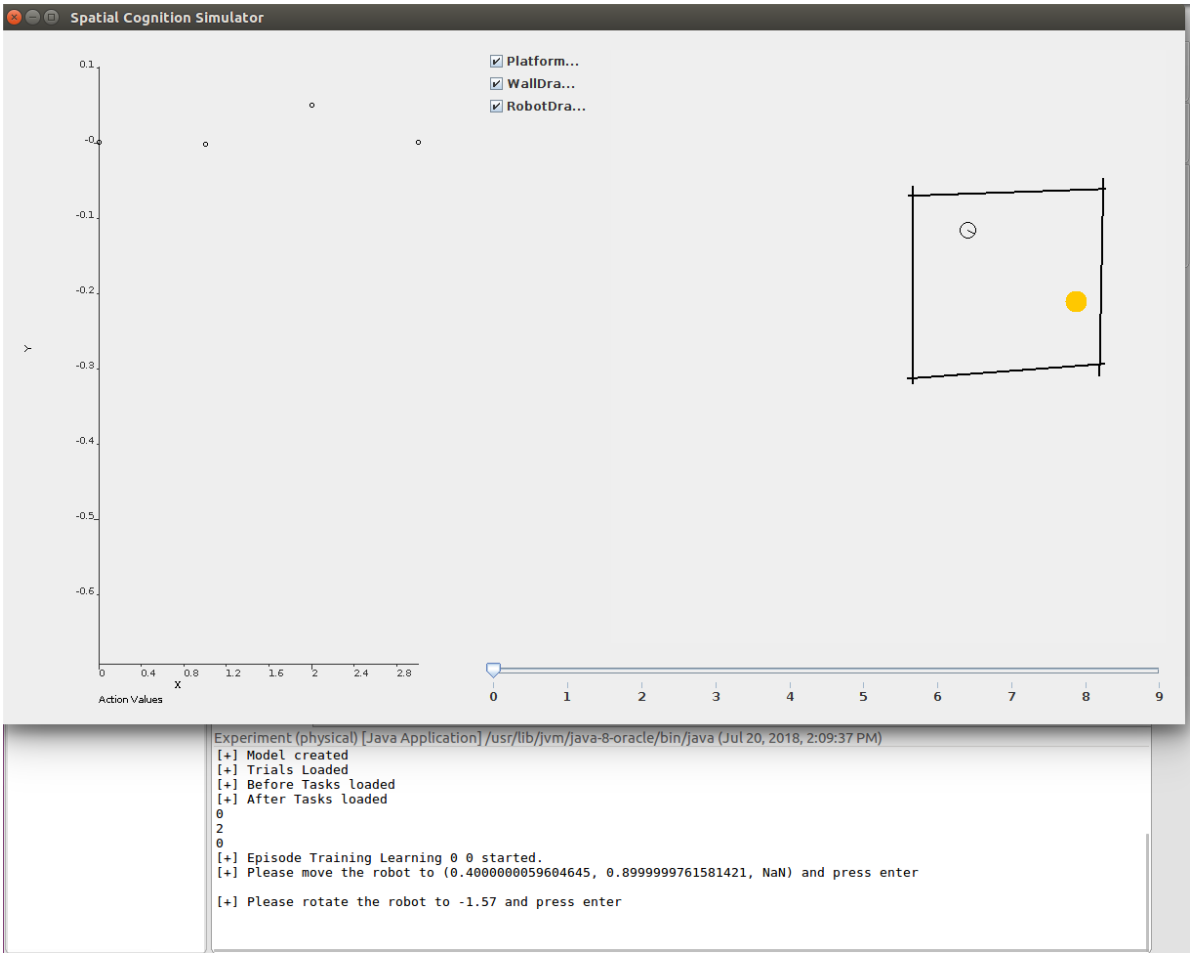

Figure 5: SCS application window during typical Robotito episode.

8. Once you see a window similar to Figure 5, the application is successfully running.
9. (Optional) Placing internal walls
    a. Some work was performed to create obstacles. These are essentially internal walls, detected using markers.
    b. Nothing needs to be done to add such walls from the software side. If you wish to add obstacles, simply place them in the camera's field of vision and place a marker on them such that it falls on the middle of one side. You can even do this between episodes, though that may cause them remain undetected for a few cycles.
    c. Parameters for these are based in the ROSWallDetector.java. Note that the collision code currently only handles line segments, so the code here simply creates a pair of parallel walls (as if using a thin block or internal wall). This is functionally a rectangle, if the width of the object is small. It would be easy to add extra line segments to create a proper square or rectangle, but other shapes would require significant changes.