Code Modified during SCS-Robotito Extension

David Ehrenhaft

University of South Florida

In SCS, branch “v2-david”:

multiscalemodel/src/edu/usf/ratsim/robot/robotito/Robotito.java

This contains the bulk of the modifications. The Robotito now implements the “Affordance Robot” interface, including all methods necessary to check the validity of each affordance and to execute them. The “Wall Robot” interface was also implemented. Multiple methods were made synchronous to prevent errors. Multiple rotation and forward step methods were designed and tested. A calibration method for the forward steps was designed but should be linked with a persistent configuration file to improve utility.

multiscalemodel/src/edu/usf/ratsim/nsl/modules/actionselection/NoExploration.java

Platform module used in the Actor Critic model to determine the next action to perform. Unnecessary code was commented out, and a variable was added to track the number of cycles since a forward action was taken. After this value is at least 50, the selected action will be a forward as long as it is valid. This was meant to prevent any infinite turning loops, when I first believed that cause to be negative votes for forward actions, but the cause of the issue can probably be attributed to wall closeness, marker overhang, or camera noise and has been mitigated or detailed in the main report. As such, the patch may not actually resolve any issues, but it probably doesn’t cause any harm.

multiscalemodel/src/edu/usf/ratsim/robot/robotito/ROSPoseDetector.java

Made methods in the program synchronous, as there is a listener thread within. Also added a simple method to provide the position of the robot without noise spikes. This was used when creating a calibration method for the forward steps. Ultimately, this is too simplistic to be a helpful filter, and was only maintained because it wasn’t causing harm and is currently unused by any threads of execution.

multiscalemodel/src/edu/usf/ratsim/robot/robotito/ROSWallDetector.java

Made methods in this code synchronous, as there is a listener thread within. Modified constants for wall sizes and added code to the listener thread to read is AR marker values over 4 as an obstacle. This created a pair of parallel walls (line segments) that represent a block or wall inside the maze. If obstacles of larger width are needed, this will need to produce true rectangles (which would be easy to do). If non-rectangle obstacles are needed, substantial modifications to class and the Robotito.java class will be necessary.

platform/src/edu/usf/experiment/task/platform/CreatePlatformROS.java

Created a class to publish a platform position using the XML file. This is necessary for the ROS approach, as the maze itself is not based on a file as with the virtual robot. Modifications may be needed to use this task with a virtual robot, where a platform is already generated by the maze file.

multiscalemodel/src/edu/usf/ratsim/model/morris/multiscale/Robotito_Experiment.xml

Created this file to act as the main XML for storing Robotito experiment parameters. This file should be used as a model to create other XML files, if necessary.

multiscalemodel/src/edu/usf/ratsim/model/morris/multiscale/ROSBasicTest.xml

Created this simplified test XML file. Identical to Robotito_Experiment.xml file but has fewer episodes per trial.

multiscalemodel/src/edu/usf/ratsim/model/morris/multiscale/obstacles/virtualObsBeforeRecall.xml and **multiscalemodel/src/edu/usf/ratsim/model/morris/multiscale/obstacles/virtualObsBeforeTraining.xml**

Created these test XML files meant for the virtual robot. These add wall obstacles to the simulated maze either before or after training. These went unused but may be helpful when performing obstacle testing.

platform/src/edu/usf/experiment/robot/CalibratableRobot.java and

platform/src/edu/usf/experiment/task/robot/CalibrateRobot.java

Created these files while experimenting with calibration. These are a very simple way to get a robot to calibrate anything it needs to as a task before each episode or trial. May be helpful eventually but the method might need to generate a calibration file to save results.

In Robotito, branch SCSCompatible:

robotito/src/pose_publisher.py and **robotito/src/wall_publisher.py**

These files initially crashed due to syntax errors – the code was trying to access the parent frames in the tf library to identify which camera current saw the robot/a particular wall, and if it was identified by both, to select only one camera's position estimate. The reasoning was fine, but the approach to identify the camera caused a syntax error. To simplify the task, I removed the second camera and overlap testing. This will need to be reimplemented in a new way to add in the second camera.

robotito/launch/pose_detector.launch

Removed second Stringray camera to simplify testing. The original ROS launch file is renamed to "robotito/launch/pose_detector_two_cameras.launch".

robotito/arduino/firmware/MotorManager.h

Originally, if the Xbee didn't receive packets for 4 seconds, the robot would release its motors without providing a way of reengaging them. I modified the file to simply engage the motors whenever a new packet is sent, preventing the Robotito from stopping indefinitely.

robotito/cfg/stingray1.yaml

Modified file to improve camera picture quality at expense of framerate. This exchange was acceptable because the AR marker detector is still slower than the resulting framerate and the marker detector is notably more effective with higher-quality images.