

COMP-4510-Project-3

This file contains logic to construct and test a navigation system for wall following generated through reinforcement learning.

On a high level, this either exploits or finds a mapping between robot states and actions, that result in some consistent or learned behavior.

Intended is to learn a type of wall following behavior that prevents the robot from crashing while maintain a constant distance to a wall on the right of the robots frame.

Table Of Contents

- /launch/wallfollow.launch
 - Launch file containing model and world information
- /src/q_learning.py
 - Primary navigation software. - Is used to train new RL models, either using SARSA or Q learning - Demos Q tables in simulation
- /src/Optimal_Q_Table_TD.JSON
 - The best Q table for Temporal Difference Learning (try demoing)
- /src/Optimal_Q_TABLE_SARSA.JSON
 - The best Q table for SARSA learning (try demoing)
- /src/known_states_tracker.JSON
 - List of states and actions that are used to track the behavior during training.
 - Informs the learning convergence plots.
- /src/Test_Q_table.JSON
 - File placeholder for training throwaway Q tables (You can write over this)

Watch the demo of the robot navigating

[!\[Watch the video\]](#)

Run Setup Files

First in its own terminal start the launch file.

```
roslaunch wallfollowing wallfollow.launch
```

If this throws an error, you may need to resource the terminal

```
cd catkin_ws
source devel/setup.bash
roslaunch wallfollowing wallfollow.launch
```

Navigation Software

The file

```
/src/q_learning.py
```

Here you can train a new model, or demo a pre-saved Q table (behavior)

Run

```
roslaunch wallfollowing q_learning.py --help
```

For more information about how to start training/testing cycles.

Training a new RL model

The simplest way to train a new model is with

```
roslaunch wallfollowing q_learning.py --train
```

This will launch a training cycle with all default parameters.
However, it is more useful to specify some of your own parameters.
Try running,

```
roslaunch wallfollowing q_learning.py --train --num_epochs=100 --out_filename Test_Q_table --plot_out_file 'Default Plots' --strategy 'TD'
```

This will launch a training cycle for 100 episodes, and save the final q table to the file 'Test_Q_table'
Note, all files are saved to the file location where the .py script is running and will write over any existing files. (Run carefully)
Code Breakdown

- num_epochs <--- Number of episodes in a learning cycle
- out_filename <--- File name to save Q table
- plot_out_file <--- file name to save convergence plots
- strategy <--- this is a mode section that can be 'Temporal Difference' or 'SARSA'

Testing a model

Here the behavior of a Q table is tested in simulation,
The Q table is not updated during this mode.
The fastest way to demo a Q table is to run,

```
roslaunch wallfollowing q_learning.py --demo
```

This will automatically select the optimal Q table for temporal difference and demo it over 25 cycles.
However, you can also select a different Q table.
run

```
roslaunch wallfollowing q_learning.py --demo --in_filename 'Optimal_Q_Table_TD'
```

to demo the the best Q table for temporal difference
Run

```
roslaunch wallfollowing q_learning.py --demo --in_filename 'Optimal_Q_Table_SARSA'
```

I think the SARSA Q table has better performance.
to demo the best Q table for SARSA
Finally, run

```
roslaunch wallfollowing q_learning.py --demo --in_filename 'Test_Q_table'
```

to demo the Q table you made in the previous section.

Connecting to the Robot (Turtlebot 3 Waffle Pi)

Ensure the following is installed

```
sudo apt-get install ros-kinetic-dynamixel-sdk
sudo apt-get install ros-kinetic-turtlebot3-msgs
sudo apt-get install ros-kinetic-turtlebot3
```

Export the waffle pi model,

```
echo "export TURTLEBOT3_MODEL=waffle_pi" >> ~/.bashrc
```

SSH into the robot in its own terminal.

```
ssh ubuntu@192.168.9.{Robot Number}
```

Or turtlebot 1 and 2,

```
ssh pi@192.168.9.{Robot Number}
```

Next, run the bring up software,

```
roslaunch turtlebot3_bringup turtlebot3_robot.launch
```

In a new terminal

Find your wifi IP address under inet Addr, ###.###.###

```
ifconfig
```

Update ~/.bashrc with the correct ROS master IP

```
vim ~/.bashrc
```

The last line has the form,

```
export ROS_MASTER_URI=http://192.168.9.{Robot Number}:11311
export ROS_HOSTNAME={Your Computers Host IP}
```

For help with vim, see the help with vim section.

After updating and saving ~/.bashrc, source the terminal

```
source ~/.bashrc
```

You can try to teleop the robot with,

```
roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch
```

Or run the navigation software by following the next section.

Running on the robot

After connecting to the ROS master on the robot, run the command,

```
roslaunch wallfollowing q_learning.py --robot --in_filename Optimal_Q_Table_SARSA
```

For best result, place robot near a wall before starting.

Troubleshooting

It is likely you will need to resource every terminal you enter.

```
cd catkin_ws
source devel/setup.bash
```

Help With VIM

Start by entering insert mode by pressing 'i'.

Make the necessary changes to the document.

To save, press,

'esc'

','

'wq'

Then press 'enter'

The document is now saved, don't forget to resource the terminal after.