Project Hecate

Generated by Doxygen 1.8.15

1 README	1
2 Hierarchical Index	7
2.1 Class Hierarchy	7
3 Class Index	9
3.1 Class List	9
4 File Index	11
4.1 File List	11
5 Class Documentation	13
5.1 Navigation Class Reference	13
5.1.1 Detailed Description	14
5.1.2 Constructor & Destructor Documentation	
5.1.2.1 Navigation()	14
5.1.2.2 ~Navigation()	14
5.1.3 Member Function Documentation	15
5.1.3.1 action()	15
5.1.3.2 demoAction()	15
5.1.3.3 dom()	16
5.1.3.4 environmentReset()	16
5.1.3.5 getStateIndex()	16
5.1.3.6 runRobot() [1/2]	17
5.1.3.7 runRobot() [2/2]	17
5.2 QLearning Class Reference	18
5.2.1 Detailed Description	18
5.2.2 Constructor & Destructor Documentation	18
5.2.2.1 QLearning()	18
5.2.3 Member Function Documentation	19
5.2.3.1 chooseAction()	19
5.2.3.2 demo()	19
5.2.3.3 getEpsilon()	20
5.2.3.4 getQtable()	20
5.2.3.5 qLearn()	20
5.2.3.6 robotLearn()	21
5.2.3.7 setEpsilon()	21
5.2.3.8 setQtable()	21
5.2.3.9 testStoreQ()	23
5.3 TurtlebotStates Class Reference	23
5.3.1 Detailed Description	24
5.3.2 Constructor & Destructor Documentation	24
5.3.2.1 TurtlebotStates()	24
5.3.2.2 ~TurtlebotStates()	24

Index

5.3.3 Member Function Documentation	2	25
5.3.3.1 findLaserDepth()	2	25
5.3.3.2 flagCollision()	2	25
5.3.3.3 returnLaserState()	2	25
6 File Documentation	2	27
6.1 include/Navigation.hpp File Reference	2	27
6.1.1 Detailed Description	2	27
6.2 include/QLearning.hpp File Reference	2	28
6.2.1 Detailed Description	2	28
6.3 include/TurtlebotStates.hpp File Reference	2	29
6.3.1 Detailed Description	2	29
6.4 src/Navigation.cpp File Reference	3	30
6.4.1 Detailed Description	3	30
6.5 src/QLearning.cpp File Reference	3	31
6.5.1 Detailed Description	3	31
6.6 src/TurtlebotStates.cpp File Reference	3	32
6.6.1 Detailed Description	3	32
6.7 test/NavigationTest.cpp File Reference	3	33
6.7.1 Detailed Description	3	33
6.7.2 Function Documentation	3	34
6.7.2.1 TEST() [1/3]	3	34
6.7.2.2 TEST() [2/3]	3	34
6.7.2.3 TEST() [3/3]	3	34
6.8 test/QlearningTest.cpp File Reference	3	35
6.8.1 Detailed Description	3	35
6.8.2 Function Documentation	3	36
6.8.2.1 TEST() [1/2]	3	36
6.8.2.2 TEST() [2/2]	3	36
6.9 test/TurtlebotstatesTest.cpp File Reference	3	36
6.9.1 Detailed Description	3	37
6.9.2 Function Documentation	3	37
6.9.2.1 TEST() [1/2]	3	37
6.9.2.2 TEST() [2/2]	3	38

39

Chapter 1

README

project_hecate

Overview

We propose a self-navigating package delivery robot, capable of finding route between logistic stations and deliver mobile parts like electric circuits, motherboards, screens and similar embedded parts from the manufacturing unit to the assembly line, in large factory units, like the Apple's factory in China. Such autonomous robotic system with inherent artificial intelligence to find it's way in factories and avoid collisions while traversing, has been developed to yield big returns to Acme robotics.

Main features of the product

- · Capable of 'learning to find it's way' in a factory/random environment
- · Obstacle avoidance
- Spawns at its default location (origin in the gazebo world) and when user commands to deliver a package, it moves to Point A to collect the package. It waits for the factory worker to put the package on it for 5 seconds and then moves towards the Point B, to deliver the package.
- · Autonomous navigation

System Design and Algorithm

The architecture involves a turtlebot which has laserscan and odometry to receive the "state" of its environment and actuator control system which allows the turtlebot to take three actions, including move straight, turn right and turn left

The algorithm implemented is called reinforcement learning (RL). RL is an aspect of Machine learning where an agent learns to behave in an environment, by performing certain actions and observing the rewards/results which it gets from those actions. It's important to note that while RL at its core aims to maximize rewards/gains, implementing a greedy approach, doesn't always lead to successful learning.

2 README

Demo Steps

Build Steps

```
cd mkdir -p '/catkin_ws/src
cd /catkin_ws/
catkin_make
cd src
git clone https://github.com/ToyasDhake/project_hecate.git
cd ...
catkin_make
```

Demo Steps

The user has to specify two points in the gazebo world-

- 1. Point A- This is the point the turtleboit navigates to, from the origin resting place,in order to receive the load package from the factory worker. The turtlebot waits for the factory worker for about 5 seconds to put on the load. (syntax: xInitial:= X Coordinate of Point A yInitial:= Y Coordinate of Point A)
- 2. Point B- This is the point the turtlebot navigates to, after picking up the load from Point A, to drop the load at Point B. (syntax: xFinal:= X Coordinate of Point B yFinal:= Y Coordinate of Point B For example, in the commands below, Point A coordinates is (2,2) and Point B coordinates is (0,7). With our experiments we found that this is one of the tough combinations for the RL to predict trajectory of the turtlebot, but our results are pretty good even on these points.

```
source devel/setup.bash
#To load Default RL trained model
roslaunch project_hecate testHecate.launch xInitial:=2 yInitial:=2 xFinal:=0 yFinal:=7
# To train a custom model
roslaunch project_hecate trainHecate.launch path:=<path_to_save>
Note : <path_to_save> should be an absolute path. For example:
"/home/shivam/catkin_ws/src/project_hecate/model.csv"
#To load custom RL model trained by the user
roslaunch project_hecate testHecate.launch xInitial:=2 yInitial:=2 xFinal:=0 yFinal:=7 path:=<path_to_table>
```

Test Steps

cd /catkin_ws/
catkin_make run_tests

Doxygen Steps

```
sudo apt install doxygen
cd cd cprescriptcd cygen -g
doxygen
cd latex
make
```

Creates a pdf containing doxygen documentation. The same can be found in the repository under the Documentation folder.

Rosbag record and Play

To record:

```
cd /catkin_ws/
source devel/setup.bash
roslaunch project_hecate testHecate.launch xInitial:=2 yInitial:=2 xFinal:=0 yFinal:=7 rosbagEnable:=true
```

To play: Download the rosbag from the following link: https://drive.google.com/open?id=1m↔ X1fIqaXv8HXWAwkDrOgJz17dvpblmXe and place it in the results folder of this repository

Terminal 1:
roscore

roscore
Terminal2:
cd /catkin_ws/src/project_hecate/results
rosbag play hecate.bag

Dependencies

ROS Kinetic

TurtleBot v2

ROS Kinetic

Gazebo 7.4 and above

Catkin

Results

The following gif shows the training of the turtlebot to "learn its way" through a floor map. During training as shown below, the turtlebot starts from the origin and then tries to navigate by taking actions of - going straight, take a left or take a right, in each episode. For each of these three actions, the turtlebot receives a reward. An episode involves a set of actions till the turtlebot collides. The episode ends after collision. The priciple during training is to achieve maximum sum of rewards in an episode. With more epochs of training, the turtlebot tries to maximize its rewards in the episode and stores the actions it took for the given states, which led to it earning maximum episode rewards.

During Inference, the turtlebot uses its learnt knowledge during training to decide on what actions to take, given a state, which had earlier fetched it maximum rewards during training. This is illustrated in the gif below, where the turtlebot is given Point A= (2,2) and Point B= (0,7). When the turtlebot starts from the origin to move towards (2,2), it attains maximum rewards for taking the action of turning slightly right towards (2,2) and then move straight towards it. Upon reaching point B, the turtlebot waits for the factory worker to place the weigth on the turtlebot. Again, when the turtlebot starts moving towards (0,7), for each small movement, the turtlebot looks inside the trained model to understand what action it should take based on what action earned it maximum reward when it was in similar state during training. Accordingly, the turtlebot finds it's way to the desired point B.

Assumptions:

- 1. We assume that the gazebo world is not changed drastically. Although the RL algorithm is capable of performing well in a dynamic world it was not trained on, drastic changes may require hyperparameter tuning of the algorithm.
- 2. We train the model on the gazebo simulator and assume that it performs well on real world too.
- 3. Acme Robotics has powerful systems with Ubuntu 16 and Ros kinetics with Gazebo (I7 processor, 16 GB RAM).
- 4. We assume that the obstacles are stationary.

Known Issues and Limitations

- 1. The RL algorithm is under active research. The algorithm implemented navigates the robot autonomously and collision free from point A to Point B, but ocassionally the path taken is not highly optimized.
- 2. The training of the turtlebot is highly compute intensive.
- The Reinforcement learning algorithm was developed with hyperparametrs optimized for the gazebo world used in the simulation. New worlds may requires training the RL world with hyperparameter tuning and modifications.
- 4. The user has to define the Point A and Point B within the rectangular walls of the gazebo world. If not done so, the turtlebot would go towards the wall to reach the point, then avoid it and go back again and repeat in a loop. 5 The Cpplint forbids the use of "non-const reference". But passing "const" to ROS function callbacks is not allowed.

4 README

Developer Documentation

- To train the model on a new gazebo world, tune the hyperparamers like the epsilon value, rewards. The
 developer might have to experiment with the linear and angular velocities for the robot to move take actions
 slower for the RI states for better training.
- 2. Train the model with good number of epochs.
- 3. Create the gazebo environment as a single model instead of agreegating seprate models and building the environment. This is because when we call reset Environment in the training pipeline, the orientation of objects resets to (0,0,0).

Agile Development Process details

We followed the Pair Programming development procedure which started with both the contributors doing an extensive literature survey. After agreement on the algorithm and the approach, we moved ahead with the first level iteration planning. The week one of the project involved development of stub functions, which was Toyas (driver) while Shivam (navigator) was involved in planning and sanity check as well as a prototype development for the complete product as a proof of concept. The week 2 was started with the review of the potential risks and the remaining backlogs. We switched the roles, and the current driver implemented the Reinforcement learning algorithm, while the navigator did the review and to ensure quality of the product delivery. This development, implementation and testing was continued in the Iteration 3 of the project.

Documentation

Product Backlog and Sprint Schedule

The product backlog file can be accessed here

The Sprint planning and review document can be accessed here

The presentation is available here

License

BSD 3-Clause License Copyright (c) 2019, Shivam Akhauri, Toyas Dhake All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: 1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. 3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Contributors

Shivam Akhauri

Former Artificial Intelligence Engineer at Ether Labs. -Former Machine learning Engineer and Project Lead at Tata Elxsi. -Skilled in Al/ML with applications in Computer vision, NLP and Robotics.

Toyas Dhake

Robotics engineer, University of Maryland College Park. -Skilled in embedded system with applications involving Arduino, Raspberry Pi and Jetson Boards.

6 README

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

QLearning	18
TurtlebotStates	23
Navigation	13

8 Hierarchical Index

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Navigation		
	Class Navigation This class contains members to generate linear and angular velocities to the urtulebot based on the depth from the obstacle information received from the depthCalculator	13
QLearning		
С	Class Qlearning class to perform reinforcement learning algorithm	18
TurtlebotSta	tates	
	Class depthCalculatio This class contains members to calculate distance for the objects which sobtained from laserscan topic. It also contains members to raise a flag if about to collide	23

10 Class Index

Chapter 4

File Index

4.1 File List

Here is a list of all documented files with brief descriptions:

include/Navigation.hpp	
Header for the robot autonomous of the robot	27
include/QLearning.hpp	
Header for the RL algorithm implementation	28
include/TurtlebotStates.hpp	
Header for reading the robot current states	29
src/Navigation.cpp	
Code for autonoumous naigation of the robot from a user defined start point to a stop point	30
src/QLearning.cpp	
Code to define the reinforcement learning pipeline	31
src/TurtlebotStates.cpp	
Code to read the current states of the turtlebot	32
test/NavigationTest.cpp	
Test cases for class Navigation	33
test/QlearningTest.cpp	
Test cases for class Qlearning	35
test/TurtlebotstatesTest.cpp	
Test cases for class Turtlebotstates	36

12 File Index

Chapter 5

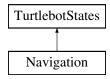
Class Documentation

5.1 Navigation Class Reference

Class Navigation This class contains members to generate linear and angular velocities to the turtulebot based on the depth from the obstacle information received from the depthCalculator.

```
#include <Navigation.hpp>
```

Inheritance diagram for Navigation:



Public Member Functions

· Navigation ()

constructor Navigation class

• ∼Navigation ()

destructor Navigation class

void runRobot (double ix, double fx, double fy, QLearning &qLearning, std::vector< int > state, ros::Rate loop_rate)

function runRobot

void runRobot (std::string path, int &highestReward, int &episodeCount, int totalEpisode, int &nextStateIndex, ros::Rate loop_rate, int innerLoopLimit)

function runRobot

• int getStateIndex (std::vector< int > state)

function getStateIndex

• void action (int action, bool &colStatus, int &reward, int &nextState)

function action

• void environmentReset ()

function environmentReset

· void demoAction (int action)

function demoAction

void dom (const nav_msgs::Odometry::ConstPtr &msg)

function dom

14 Class Documentation

Public Attributes

- double x
- double y
- double **z**
- double roll
- double pitch = 0
- · double yaw
- double x_goal
- · double y_goal

5.1.1 Detailed Description

Class Navigation This class contains members to generate linear and angular velocities to the turtulebot based on the depth from the obstacle information received from the depthCalculator.

5.1.2 Constructor & Destructor Documentation

5.1.2.1 Navigation()

Navigation::Navigation ()

constructor Navigation class

Parameters

none

Returns

none initializes the publisher and subsciber initialize the value of odometry initialize the liner and angular speed

5.1.2.2 \sim Navigation()

Navigation:: \sim Navigation ()

destructor Navigation class

Parameters

none

Returns

none Destructor for the navigation clas

5.1.3 Member Function Documentation

5.1.3.1 action()

```
void Navigation::action (
    int action,
    bool & colStatus,
    int & reward,
    int & nextState )
```

function action

Parameters

int	action
bool	&colStatus
int	&reward
int	&nextState

Returns

none publishes linear and angular velocities to the turtlebot

5.1.3.2 demoAction()

function demoAction

Parameters

```
int action
```

Returns

none publishes linear and angular velocities to the turtlebot

16 Class Documentation

```
5.1.3.3 dom()
```

function dom

Parameters

```
const | nav_msgs::Odometry::ConstPtr
```

Returns

none callback to read odometry

5.1.3.4 environmentReset()

```
void Navigation::environmentReset ( )
```

function environmentReset

Parameters

none

Returns

none resets the gazebo environment

5.1.3.5 getStateIndex()

```
int Navigation::getStateIndex ( std::vector < \ int \ > \ state \ )
```

function getStateIndex

Parameters

```
std::vector<int> state
```

Returns

int stateIndex mapping the vector to the state in rl table

5.1.3.6 runRobot() [1/2]

function runRobot

Parameters

double	ix
double	fx
double	fy
QLearning	&qLearning
std::vector <int></int>	state
ros::Rate	loop_rate

Returns

none Runs the inferece code the bot uses the trained model to navigate

5.1.3.7 runRobot() [2/2]

```
void Navigation::runRobot (
    std::string path,
    int & highestReward,
    int & episodeCount,
    int totalEpisode,
    int & nextStateIndex,
    ros::Rate loop_rate,
    int innerLoopLimit )
```

function runRobot

Parameters

std::string	path
int	&highestReward
int	&episodeCount
int	totalEpisode, int &nextStateIndex
ros::Rate	loop_rate
int	innerLoopLimit

18 Class Documentation

Returns

none training of the agent by receiving states perform actions in that states and receive rewards

The documentation for this class was generated from the following files:

- · include/Navigation.hpp
- src/Navigation.cpp

5.2 QLearning Class Reference

Class Qlearning class to perform reinforcement learning algorithm.

```
#include <QLearning.hpp>
```

Public Member Functions

• QLearning ()

constructor Qlearning class

• void setEpsilon (double e)

function setEpsilon

double getEpsilon ()

function getEpsilon

void setQtable (std::string path)

function setQtable

void getQtable (std::string path)

function getQtable

• void qLearn (int state, int action, int reward, double val)

function qlearn

void robotLearn (int si, int act, int rew, int nsi)

function robotLearn

· void testStoreQ ()

function testStoreQ

• int demo (int index, bool collision, double angleToGoal)

function demo

• int chooseAction (int index)

function chooseAction

5.2.1 Detailed Description

Class Qlearning class to perform reinforcement learning algorithm.

5.2.2 Constructor & Destructor Documentation

5.2.2.1 QLearning()

```
QLearning::QLearning ( )
```

constructor Qlearning class

Parameters

none

Returns

none intililizes the reinforcement learning model

5.2.3 Member Function Documentation

5.2.3.1 chooseAction()

function chooseAction

Parameters

```
int index
```

Returns

int action robots action selection for the state

5.2.3.2 demo()

```
int QLearning::demo (
    int index,
    bool collision,
    double angleToGoal )
```

function demo

Parameters

int	index
bool	collision
double	angleToGoal

Returns

int action use the rl model to decide the best action

20 Class Documentation

5.2.3.3 getEpsilon()

```
double QLearning::getEpsilon ( )
```

function getEpsilon

Parameters

none

Returns

double epsilon as getter for epsilon

5.2.3.4 getQtable()

function getQtable

Parameters

```
std::string path
```

Returns

none loads the pretrained RL model

5.2.3.5 qLearn()

function qlearn

Parameters

int	state
int	action
int	reward
double	val

Returns

none updates reinforcement learning model

5.2.3.6 robotLearn()

```
void QLearning::robotLearn (
    int si,
    int act,
    int rew,
    int nsi )
```

function robotLearn

Parameters

int	si
int	act
int	rew
int	nsi

Returns

none applies the boltzmann equation to apply RL

5.2.3.7 setEpsilon()

function setEpsilon

Parameters



Returns

none setter for epsilon

5.2.3.8 setQtable()

22 Class Documentation

function setQtable

Parameters

std::string	path
-------------	------

Returns

none stores the rl model

5.2.3.9 testStoreQ()

```
void QLearning::testStoreQ ( )
```

function testStoreQ

Parameters

none

Returns

none function for inference quality test of the rl model

The documentation for this class was generated from the following files:

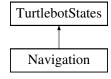
- include/QLearning.hpp
- src/QLearning.cpp

5.3 TurtlebotStates Class Reference

Class depthCalculatio This class contains members to calculate distance for the objects which is obtained from laserscan topic. It also contains members to raise a flag if about to collide.

```
#include <TurtlebotStates.hpp>
```

Inheritance diagram for TurtlebotStates:



24 Class Documentation

Public Member Functions

• TurtlebotStates ()

constructor TurtlebotStates

∼TurtlebotStates ()

destructor TurtlebotStates

void findLaserDepth (const sensor_msgs::LaserScan::ConstPtr &msg)

function findLaserDepth

• bool flagCollision ()

function flagCollision

• std::vector< int > returnLaserState ()

function returnLaserState()

5.3.1 Detailed Description

Class depthCalculatio This class contains members to calculate distance for the objects which is obtained from laserscan topic. It also contains members to raise a flag if about to collide.

5.3.2 Constructor & Destructor Documentation

5.3.2.1 TurtlebotStates()

TurtlebotStates::TurtlebotStates ()

constructor TurtlebotStates

Parameters

none

Returns

none initializes the collisionStatus flag

5.3.2.2 \sim TurtlebotStates()

TurtlebotStates::~TurtlebotStates ()

destructor TurtlebotStates

Parameters

none

Returns

none destroy the TurtlebotStates

5.3.3 Member Function Documentation

5.3.3.1 findLaserDepth()

function findLaserDepth

Parameters

```
msg type sensor_msgs::LaserScan
```

Returns

none function to read LaserScan sensor messages and raise flag if distance of the obstacle is less than threshold

5.3.3.2 flagCollision()

```
bool TurtlebotStates::flagCollision ( )
```

function flagCollision

Parameters

none

Returns

1 if very close to obstacle and 0 if not close Return the current value of collisionStatus

5.3.3.3 returnLaserState()

```
std::vector< int > TurtlebotStates::returnLaserState ( )
```

function returnLaserState()

26 Class Documentation

D _o			- 4		
Pа	ra	m	eı	e	rs

none

Returns

std::vector<int> return the states for the rl algorithm using the laserscan

The documentation for this class was generated from the following files:

- include/TurtlebotStates.hpp
- src/TurtlebotStates.cpp

Chapter 6

File Documentation

6.1 include/Navigation.hpp File Reference

Header for the robot autonomous of the robot.

```
#include <ros/ros.h>
#include <vector>
#include <string>
#include <iostream>
#include "sensor_msgs/LaserScan.h"
#include "std_srvs/Empty.h"
#include "geometry_msgs/Twist.h"
#include "TurtlebotStates.hpp"
#include "QLearning.hpp"
#include "nav_msgs/Odometry.h"
```

Classes

· class Navigation

Class Navigation This class contains members to generate linear and angular velocities to the turtulebot based on the depth from the obstacle information received from the depthCalculator.

6.1.1 Detailed Description

Header for the robot autonomous of the robot.

- 1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

28 File Documentation

3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission. THIS SOFTWARE IS PROV
□ IDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILI
□ TY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPY
□ RIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT
OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERR
□ UPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE
USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Author

Shivam Akhauri, Toyas Dhake

Date

27 November 2019

Copyright

BSD 3-clause, 2019 Shivam Akhauri, Toyas Dhake

6.2 include/QLearning.hpp File Reference

Header for the RL algorithm implementation.

```
#include <iostream>
#include <algorithm>
#include <vector>
#include <string>
#include <array>
```

Classes

class QLearning

Class Qlearning class to perform reinforcement learning algorithm.

6.2.1 Detailed Description

Header for the RL algorithm implementation.

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission. THIS SOFTWARE IS PROV
□ IDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILI
□ TY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPY
□ RIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT
OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERR
□ UPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE
USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Author

Shivam Akhauri, Toyas Dhake

Date

27 November 2019

Copyright

BSD 3-clause, 2019 Shivam Akhauri, Toyas Dhake

6.3 include/TurtlebotStates.hpp File Reference

Header for reading the robot current states.

```
#include <vector>
#include "ros/ros.h"
#include "geometry_msgs/Twist.h"
#include "sensor_msgs/LaserScan.h"
```

Classes

class TurtlebotStates

Class depthCalculatio This class contains members to calculate distance for the objects which is obtained from laserscan topic. It also contains members to raise a flag if about to collide.

6.3.1 Detailed Description

Header for reading the robot current states.

- 1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

30 File Documentation

3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission. THIS SOFTWARE IS PROV
□ IDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILI
□ TY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPY
□ RIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT
OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERR
□ UPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE
USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Author

Shivam Akhauri, Toyas Dhake

Date

27 November 2019

Copyright

BSD 3-clause, 2019 Shivam Akhauri, Toyas Dhake

6.4 src/Navigation.cpp File Reference

Code for autonoumous naigation of the robot from a user defined start point to a stop point.

```
#include <tf/transform_listener.h>
#include <cmath>
#include <boost/range/irange.hpp>
#include "Navigation.hpp"
```

6.4.1 Detailed Description

Code for autonoumous naigation of the robot from a user defined start point to a stop point.

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- 3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission. THIS SOFTWARE IS PROV
 □ IDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILI
 □ TY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPY
 □ RIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
 EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT
 OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERR
 □ UPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE
 USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

```
Author
Shivam Akhauri, Toyas Dhake

Date
27 November 2019

Copyright
BSD 3-clause, 2019 Shivam Akhauri, Toyas Dhake
```

6.5 src/QLearning.cpp File Reference

Code to define the reinforcement learning pipeline.

```
#include <time.h>
#include <ros/ros.h>
#include <iostream>
#include <sstream>
#include <fstream>
#include <vector>
#include <cmath>
#include <cmath>
#include <cstdlib>
#include <random>
#include <boost/range/irange.hpp>
#include "QLearning.hpp"
```

6.5.1 Detailed Description

Code to define the reinforcement learning pipeline.

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- 3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission. THIS SOFTWARE IS PROV← IDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILI← TY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPY← RIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERR← UPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

32 File Documentation

Author

Shivam Akhauri, Toyas Dhake

Date

27 November 2019

Copyright

BSD 3-clause, 2019 Shivam Akhauri, Toyas Dhake

6.6 src/TurtlebotStates.cpp File Reference

Code to read the current states of the turtlebot.

```
#include <iostream>
#include <cfloat>
#include <cmath>
#include "TurtlebotStates.hpp"
#include <boost/range/irange.hpp>
```

6.6.1 Detailed Description

Code to read the current states of the turtlebot.

BSD 3-Clause License Copyright (c) 2019, Shivam Akhauri, Toyas Dhake All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- 3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission. THIS SOFTWARE IS PROV
 □ IDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILI
 □ TY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPY
 □ RIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
 EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT
 OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERR
 □ UPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE
 USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Author

Shivam Akhauri, Toyas Dhake

Date

27 November 2019

Copyright

BSD 3-clause, 2019 Shivam Akhauri, Toyas Dhake

6.7 test/NavigationTest.cpp File Reference

Test cases for class Navigation.

```
#include <gtest/gtest.h>
#include <ros/ros.h>
#include <geometry_msgs/Pose.h>
#include <geometry_msgs/Twist.h>
#include <vector>
#include "Navigation.hpp"
#include "QLearning.hpp"
```

Functions

TEST (TESTNavigation, checkForCorrectStateIndex)

Tests to verify the correctness of the state indices.

TEST (TESTNavigation, checkForTestRobot)

Tests to verify if the turtlebot reaches the desired loaction.

TEST (TESTNavigation, checkForTrainRobot)

Tests to verify if the RI algorithm is training as expected.

6.7.1 Detailed Description

Test cases for class Navigation.

BSD 3-Clause License Copyright (c) 2019, Shivam Akhauri, Toyas Dhake All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- 3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission. THIS SOFTWARE IS PROV← IDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILI← TY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPY← RIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERR← UPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Author

Shivam Akhauri, Toyas Dhake

Copyright

3-clause BSD

34 File Documentation

6.7.2 Function Documentation

Tests to verify the correctness of the state indices.

Parameters

Navigation	gtest framework
checkForCorrectStateIndex	Name of the test

Returns

none

Tests to verify if the turtlebot reaches the desired loaction.

Parameters

Navigation	gtest framework
checkForTestRobot	Name of the test

Returns

none

Tests to verify if the RI algorithm is training as expected.

Parameters

Navigation	gtest framework
checkForTrainRobot	Name of the test

Returns

none

6.8 test/QlearningTest.cpp File Reference

Test cases for class Qlearning.

```
#include <gtest/gtest.h>
#include <ros/ros.h>
#include "QLearning.hpp"
```

Functions

• TEST (TESTQlearning, testChooseAction)

Test to verify expected turtlebot action.

TEST (TestQlearning, testActionfromTheDemoFunctions)

Test to verify if demo is taking proper decision based on values in table.

6.8.1 Detailed Description

Test cases for class Qlearning.

BSD 3-Clause License Copyright (c) 2019, Shivam Akhauri, Toyas Dhake All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- 3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission. THIS SOFTWARE IS PROV
 □ IDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILI
 □ TY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPY
 □ RIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
 EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT
 OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERR
 □ UPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE
 USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Author

Shivam Akhauri, Toyas Dhake

Copyright

3-clause BSD

36 File Documentation

6.8.2 Function Documentation

Test to verify expected turtlebot action.

Parameters

TESTQlearning	gtest framework
testChooseAction	Name of the test

Returns

none

Test to verify if demo is taking proper decision based on values in table.

Parameters

TESTQlearning	gtest framework
testActionfromTheDemoFunctions	Name of the test

Returns

none

6.9 test/TurtlebotstatesTest.cpp File Reference

Test cases for class Turtlebotstates.

```
#include <gtest/gtest.h>
#include <ros/ros.h>
#include <sensor_msgs/LaserScan.h>
#include "TurtlebotStates.hpp"
```

Functions

TEST (TESTTurtlebotState, checkObstacleDetection)

Test to verify if Obstacle detection is happening properly. Obtain laserscan sensor data and raise a flag if obstacle detected.

• TEST (TESTTurtlebotState, checkDefaultflagCollisionValue)

check if flag is raised when obstacle distance is very less

6.9.1 Detailed Description

Test cases for class Turtlebotstates.

BSD 3-Clause License Copyright (c) 2019, Shivam Akhauri, Toyas Dhake All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- 3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission. THIS SOFTWARE IS PROV
 □ IDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILI
 □ TY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPY
 □ RIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
 EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT
 OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERR
 □ UPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE
 USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Author

Shivam Akhauri, Toyas Dhake

Copyright

3-clause BSD

6.9.2 Function Documentation

Test to verify if Obstacle detection is happening properly. Obtain laserscan sensor data and raise a flag if obstacle detected.

38 File Documentation

Parameters

TESTTurtlebotState	gtest framework
checkObstacleDetection	Name of the test

Returns

none

check if flag is raised when obstacle distance is very less

Parameters

TESTTurtlebotState	gtest framework
checkDefaultflagCollisionValue	Name of the test

Returns

none

Index

QLearning, 20

\sim Navigation	QLearning, 18
Navigation, 14	chooseAction, 19
~TurtlebotStates	demo, 19
TurtlebotStates, 24	getEpsilon, 19
,	getQtable, 20
action	qLearn, 20
Navigation, 15	QLearning, 18
	robotLearn, 21
chooseAction	setEpsilon, 21
QLearning, 19	setQtable, 21
	testStoreQ, 23
demo	QlearningTest.cpp
QLearning, 19	TEST, 36
demoAction	1231,00
Navigation, 15	returnLaserState
dom	TurtlebotStates, 25
Navigation, 15	robotLearn
, , , , , , , , , , , , , , , , , , ,	QLearning, 21
environmentReset	runRobot
Navigation, 16	Navigation, 16, 17
, , , , , , , , , , , , , , , , , , ,	Navigation, 16, 17
findLaserDepth	setEpsilon
TurtlebotStates, 25	QLearning, 21
flagCollision	setQtable
TurtlebotStates, 25	QLearning, 21
	src/Navigation.cpp, 30
getEpsilon	src/QLearning.cpp, 31
QLearning, 19	÷ ,,
getQtable	src/TurtlebotStates.cpp, 32
QLearning, 20	TEST
getStateIndex	NavigationTest.cpp, 34
Navigation, 16	QlearningTest.cpp, 36
3	TurtlebotstatesTest.cpp, 37, 38
include/Navigation.hpp, 27	• •
include/QLearning.hpp, 28	test/NavigationTest.cpp, 33
include/TurtlebotStates.hpp, 29	test/QlearningTest.cpp, 35
	test/TurtlebotstatesTest.cpp, 36
Navigation, 13	testStoreQ
\sim Navigation, 14	QLearning, 23
action, 15	TurtlebotStates, 23
demoAction, 15	~TurtlebotStates, 24
dom, 15	findLaserDepth, 25
environmentReset, 16	flagCollision, 25
getStateIndex, 16	returnLaserState, 25
Navigation, 14	TurtlebotStates, 24
runRobot, 16, 17	TurtlebotstatesTest.cpp
NavigationTest.cpp	TEST, 37, 38
TEST, 34	
1201, 07	
qLearn	
•	