Assignment #4

Total score: 100

Due date: 5/8 (Monday)

Objective: Implement a Machine Learning Method for Robot Navigation.

In this assignment, you will develop **a program** called "**hw4.cpp**" that **intelligently controls** the navigation of the **simulated robot**, "**turtle1**" using a **machine learning method**. The **boundary of** the **turtlesim window** is defined by two diagonal corners, origin, **(0, 0)** and top-right-corner, approximately at **(11, 11)** and these two points also define the **legal boundary** for turtle1.

turtle1 will start **moving** from the default location, the **center of the window**. In addition to turtle1, there will be two additional types of turtles: **T turtles** called "**T0**", "**T1**", and "**T2**" and **X turtles** called "**X0**", "**X1**", and "**X2**". Both **T** and **X turtles** will be landed on random locations **before** your program is started and may be **moving** in a direction following **certain patterns**. Both T and X turtles will be landed on **random locations** that are far enough from the location of turtle1 to make sure that turtle1 will not be unluckily captured immediately when it is started. You can assume that T and X turtles will **never** be landed on the same location.

Any **X turtle** **will capture** **turtle1** (by killing it) when it is **within the threshold distance 0.5**. In other words, any X turtle **Xi** will capture **turtle1** when **turtle1** is within the distance 0.5. The distance between two points, (x1, y1) and (x2, y2) is measured by the Euclidean distance function, $\sqrt{(x1-x2)^2+(y1-y2)^2}$. For example, if X1 is located at (x1, y1) and turtle1 is located at (x2, y2), then turtle1 will be captured if the distance between X1 and turtle1 ≤ 0.5. Likewise, **turtle1** can also **capture** a **T turtle** by approaching itself within the same threshold distance to it.

The **mission of turtle1** is to **capture all the T turtles** **without** being **captured** by any X turtle **and** stay/move within the legal boundary of the turtlesim window.

The **mission** of turtle1 will be **failed** if it is captured by any X turtle **or** it moves beyond the legal boundary of the turtlesim window that is defined by two points (0, 0) and (11, 11).

When the mission is completed, **turtle1** should **display** (1) **total traveled distance** calculated by ∑**velocity***Δ**time**, (2) **number of target turtles** captured, and (3) the **learned moving pattern** for each T turtle, and (4) **stop** at the current position.

Steps needed to test your program

(a) Open a turtlesim window using the command "rosrun turtlesim turtlesim_node".

(b) Run the test program, **hw4test.cpp** (that will be posted on the course page soon) on a separate terminal. Like hw3test.cpp, the **primary purpose** of this program is to spawn all T and X turtles, make them move in certain directions, following certain patterns, and to capture turtle1 when it is within the threshold distance to any X turtle. However, **you don't write hw4test.cpp**. This program will be given to you for testing your program.

Note: The moving patterns of T turtles may be changed when I test your program. So you have to implement a reasonably good enough learning method for turtle1 instead of hardcoding the moving patterns of T turtles given in hw4test.cpp in order to successfully accomplish the mission.

(c) Run your own program, **hw4.cpp** that will only navigate turtle1 to accomplish the mission.
Note: Your program name **should** be **hw4.cpp**.

Additional requirements
Write a **brief report** in Word format that includes all of the following:
(a) Your team name, member name(s), contact email addresses, and also the percentage contribution to this assignment if the assignment was completed by a team. (If a team cannot reach a consensus on the individual contribution, include the individual's claimed percent contribution with a brief description on specific tasks performed.)
(b) A **brief description** about your **learning method** to accomplish the mission
(c) A **pseudo code** (NOT source code) for your **learning method**. The source code should be turned in as a separate file.
(d) **A brief description of moving patterns** learned from each T turtle

Warning: Although code reuse from existing source codes on the Internet is allowed, copying code from another student or team belonging to this class is strictly prohibited. Any student or team violating this policy will receive a **ZERO** score for this assignment.

What to submit
Zip both the **report**, your program, **hw4.cpp**, and other necessary files possibly including CMakeLists.txt, package.xml, and others if any, into **ONE file** named by **your team name**. For example, if **your team** name is "**ABC**", then include all the necessary files and make one zip file called **ABC.zip**. Submit only **ONE** file per team **to Titanium** by the due date. In addition, I strongly recommend you to write the report in Word format so that I can provide feedbacks directly in the report when necessary. If a PDF format of report is submitted, no feedback will be provided.

Grading policy
Grade will be based on the quality of your team's work based on the requirements and the written report. Full credit is given to the team that followed all the instructions, accomplished the mission, and met all other requirements. Otherwise, some points may be deducted.