Assignment #3

Total score: 100

Due date: 4/3 (Monday)

Objectives: Define proper heuristic functions and implement a heuristic algorithm to solve a robot navigation problem.

In this assignment, you will develop **a program** called "**hw3.cpp**" that **heuristically controls** the navigation of the ROS **simulated turtle robot**, "**turtle1**" (default name created by rosrun command). The **boundary of** the **turtlesim window** is defined by two diagonal corners, origin, **(0, 0)** and top-right-corner, **(11, 11)**. In this assignment, these two points define the **legal boundary** of the turtlesim window for turtle1.

turtle1 should **start moving** from the **origin** of the turtlesim window that is bottom-left-corner on the window. In addition to turtle1, there will be two additional types of turtles: the **target turtles** called "**T1**", "**T2**", and "**T3**" and the **villain turtles** called "**X1**", "**X2**", "**X3**", and "**X4**". Unlike turtle1, both **target turtles** and **villain turtles** are **stationary** (not moving) and will be landed on random locations before your program is started. Moreover, target turtles and villain turtles will be landed on **random locations** that are far enough from the location of turtle1, e.g., (x, y) >= (2, 2) in the turtlesim window, to make sure that turtle1 will not be unluckily captured immediately when it is started.

The **villain turtles** will **capture turtle1** (by killing it) when it is near enough. In other words, any villain turtle **Xi** will **capture turtle1** when **turtle1** is **within a 10% range** defined by (**x ± 0.1*x, y ± 0.1*y**) from **Xi's current location** (x,y). For example, if X1 is located at (3, 5) and turtle1 is located at (3.3, 5.5) or (2.9, 4.7), then turtle1 will be captured by X1. **turtle1** can **capture** a **target turtle**, **Ti** when it is **within a 5% range of Ti's current location**, (x, y), that is **(x ± 0.05*x, y ± 0.05*y)**. For example, if turtle1 is located at (9, 9) and T1 is located at (9.45, 9.45) or (8.55, 8.55), then turtle1 can capture T1.

The **mission** of turtle1 is to **capture all the target turtles** **without** getting **captured** by any villain turtle **and** moving **out of** the window **boundary** by traveling a **minimum distance** from the starting location.

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

When the mission is completed, **turtle1** should **display** the **total** traveled **distance** calculated by ∑velocity*Δtime, **number of target turtles** captured on the screen, and **stop** moving at the current position. See the sample program, "robot_cleaner.cpp" for example distance calculation.

Tip: You can test your program by creating the target and villain turtles by either your own test program or ROS command such as "rosservice call /spawn x, y, theta turtle-name" (see the tutorial for examples). For example, you can create a turtle called "X1" with this command:
rosservice call /spawn 5, 5, 0.2 X1

Tip: You can kill a node using rosnode command, "rosservice call /kill turtle-name" or "rosnode kill node-name". You can also kill a turtle in a C++ as shown in the example program, "spawn_turtle.cpp". Or you call a ROS command in a C++ program like in this example:

```
#include <stdlib.h>
int main() {
    system("rosservice call /kill X1");
    return 0;
}
```

Note: Your program name **should** be **hw3.cpp** and you **should** use **hw3.launch file** to execute your program. See the tutorial to learn how to create a launch file and run it.

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 **heuristic** and **strategy** to accomplish the mission

(c) A **pseudo code** (NOT source code) for your heuristic and strategy. The source code should be turned in as a separate file.

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, and other necessary files including hw3.cpp, CMakeLists.txt, package.xml, hw3.launch, and others into **ONE file** named by **your team name**. Submit only **ONE** 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. For example, if **your team** name is "**ABC**", then the zip file name should be **ABC.zip**.

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.