The Robot Tournament

Semester Project

1 Introduction

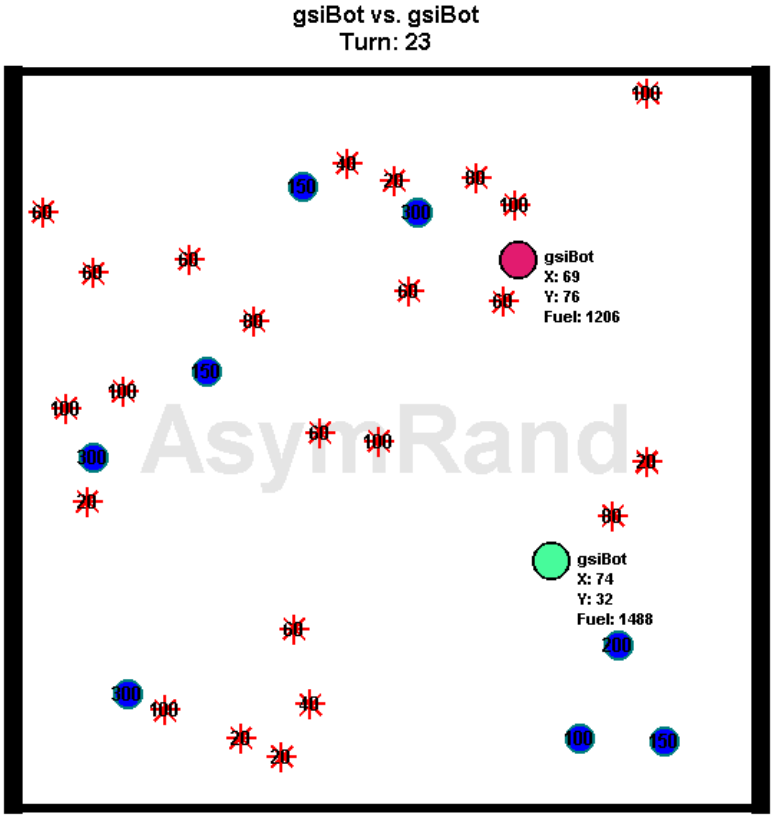
The Robot Tournament is intended to be an open ended problem designed for you to express your ingenuity and creativity through programming. For this semester project, you will be able to utilize everything you have learned about the Matlab language to program a strategy into a simulated robot. Your robot, along with those designed by your classmates, will compete against each other in a round-robin tournament. There are no right or wrong answers, only good strategies and bad strategies. Although this is a graded project, we encourage you to relax and have fun with it. By doing so, we believe you will learn a lot and have an enjoyable experience.

"*The strategy is the hardest part, not the programming...*"
- Spring 2007 Robot Tournament Winner

2 The Game

2.1 Overview

In the figure below are two agents, or robots, competing against each other in an enclosed arena. Sometimes the arena will be referred to as a map. The arena is 100 x 100 meters in size and the robots are restricted to stay within the arena throughout the game. Each robot has fuel which can be used to move around the arena. The fuel consumed will be a function of the velocity traveled. If a robot runs out of fuel, it cannot move.



Each turn, or time step, both robots will have a chance to move a small distance from their current location. Scattered around the arena are fuel tanks. Coming within a small distance of a fuel tank will increase the robots fuel level. Also, scattered around the arena are mines. Coming within a small distance of a mine will cause the robot to lose fuel.

If the robots come within a small distance from each other, then they have collided and the robot with the most fuel will be the winner.

Robots will have perfect information regarding the state of the current environment and will be remember the movement of themselves and the opponent from the previous turn.

2.2 The Rules

In this section, we will enumerate the rules precisely.

- Each turn, a robot will be able to a give a change in position, [dx, dy], and the position of the robot will be changed to (x + dx, y + dy) for the next turn where x and y coordinates of the robot.
- The speed of the robot will be limited to $\sqrt{(dx^2 + dy^2)} \leq 3$.
- Each battle will last up to 1000 turns.
- If the robots come within 5 meters of each other, then they have collided and the robot with the most fuel will be the winner.
- If no winner has been decided by the end of the battle, both robots will receive a loss.
- Robots will start with 1000 units of fuel.
- Robots will lose fuel according to the formula $dx^2 + dy^2 + 2$, where and dx and dy are the x and y-velocities, respectively.
- Robots will be stopped at the walls of the arena but will lose fuel according to the [dx, dy] values given by the robot.
- If a robot comes within 2 meters of a fuel tank, its fuel will be increased by the value of the fuel tank, and the fuel tank will be removed from the arena.
- A robot's fuel level will be capped at 1500 units.
- If a robot's fuel level drops to 0, it will not be able to move, and its fuel level will remain at 0.
- If a robot comes within 5 meters of a mine, its fuel will be decremented by the value of the mine, and the mine will be removed from the arena.

3 Your Function

3.1 Function header

You will create a function that has the following header line:

```
[out] = my_robot(self, enemy, tank, mine)
```

3.1.1 Input and Output arguments

self: structure array with fields .pos, .fuel, and .prev. The field .pos will be a 1x2 numerical array containing the (x,y) coordinates of your robot. The field .fuel will contain your robot's current fuel level. The field .prev will contain a 1x2 numerical array containing your robot's output on the previous turn.

enemy: structure array with same fields as self except with information corresponding to the enemy robot.

tank: structure array with field .pos and .val. The field .pos will contain (x,y) coordinates where the fuel tank is located. The field .val will contain the value of the fuel tank (i.e. the amount by which fuel will increase upon getting the fuel tank.

mine: structure array with same fields as tank except corresponding to the mines in the arena. The information contained in .val corresponds to the amount of fuel lost for hitting the mine.

out: a 1x2 real, double array containing the movement of your robot for this turn, i.e. an array of the form [dx, dy] where dx and dy are scalars corresponding to the change in position in the x and y directions, respectively.

3.2 Restrictions and Penalties

In the following section, we will enumerate all the possible violations during a battle and the penalties associated with them.

Throwing an Error: If your function throws an error at any time, the battle will be terminated immediately without a winner

Incorrect Outputs: If your function outputs something other than a 1x2 numerical array, the battle will be terminated immediately without a winner.

Speed Limit: If your function travels more than 3 meters in a single turn (i.e. $\sqrt{(dx^2 + dy^2)} > 3$), the battle will terminated immediately without a winner.

Time Limit: In the interest of time, your function must return a response when requested for a move within .1 seconds. Violation of this time limit will cause your robot to forfeit its move for this turn only.

4 The Tournament

4.1 Overview

The tournament will be played on set of premade maps, each with a different configuration of tanks, mines, and starting positions. You will not know the configuration of the maps ahead of time. For the tournament, every robot will battle every other robot starting from both positions (i.e. each robot will play every other robot twice) for each map. Robots will receive 1 point for every win in the tournament, 0 points for a loss, tie, or if the other robot produces an error, and -2 points for producing an error. Robots will be ranked according to the number of points earned by the end of the tournament.

The tournament will be played in two phases, Beta Testing and Main Event.

The purpose of Beta Testing is to encourage you to get an early start on your robot, to see how your robot performs in a tournament environment, and to make sure the programs running the tournament are in place before the Main Event. The Beta Test will consist of a small tournament that is representative of the Main Event tournament. You will be graded based on submission and a minimum performance level in the tournament.

The Main Event will be a large tournament consisting of approximately 500,000 matches. This tournament is where the final rankings of your robot will be determined and where most of the points for the project will be earned.

4.2 Grading

Your project grade will be out of 100 points and count for 10% of your final grade. The grade breakdown of the project is as follows.

Submitting Beta Test Robot correctly and on time:	20	pts
Having at least 1 point in the Beta Test Tournament:	10	pts
Submitting the Main Event Robot on time:	50	pts
Rankings in Main Event:	0 – 20	pts

4.3 Getting Help

If you feel like you need help understanding the tournament, please email the Head GSI at timmy.siauw@gmail.com or go to his office hours. The Head GSI will not discuss strategies or help you debug. He will only help you understand the tournament requirements so that you can turn in all the deliverables on time. This project was created so that you could do something of your own, not so that you could follow more instructions. However, if you feel lost or overwhelmed, let the Head GSI know and he will do what he can to help you get back on track. Throughout the tournament, extra tips and suggestions will be posted periodically to help you along.

4.4 Submission

Your function for the Beta Test and the Main Event must be submitted by 11:59pm on the deadline day. Submissions should be through bSpace under the assignment heading Beta Test and Main Event, respectively.

Submit your robot as an attachment m-file just as in the assignments. To identify which group each robot belongs to, there should also be an attachment .txt file called group.txt that has the following format:

group.txt

[TEAM NAME]	(max 20 characters, not necessarily same as function name)
[SIZE OF GROUP]	(max 3)
[LAST], [FIRST], [SID], [Lab #]	(one of these lines for each group member)
[LAST], [FIRST], [SID], [Lab #]	(one of these lines for each group member)
[LAST], [FIRST], [SID], [Lab #]	(one of these lines for each group member)

group.txt (example)

```
Team GSI
2
Bayen, Alex, 12345678, 0
Siauw, Timmy, 87654321, 0
```

WAYS TO LOSE POINTS

Multiple Submissions:	Only ONE person per group should submit a robot with associated .txt file. Multiple submissions will be deducted 10 points from the project grade, so please coordinate.
Display to Screen:	Suppress all lines in your function with a semicolon, and do not display anything to the screen. Robots that display to the screen will be deducted 10 points from the project grade.
Plotting:	Your robot should not plot anything or modify the game screen in any way. Robots that do so will be deducted 10 points from the project grade.
Team Name:	Any team name containing vulgarities or is inappropriate in any way will receive 0 points for the project. We will make this determination at our discretion.
While-loop:	Your function may not use a while loop. If a robot is caught having a while loop, 10 points will be deducted from the project grade. If the robot goes into an infinite loop during the Main Event, you will get a 0 for the project.
Non-Input Info:	You may not use the functions get, set, which, what, dir, global, persistent, fprintf, fopen, or any other function in an attempt to get information about the current match other than what is given as input arguments. Doing so will cause you to get a 0 for the project.

Tournament details and submission guidelines are subject to change. Any changes will be announced on bSpace. You will be responsible for the contents of these announcements so please read them.

5 Schedule

Fri, Feb 25, 2011	Prompt and relevant functions posted.
Fri, March 18, 20011	Beta Test robots due at 11:59pm
Mon, March 28, 2011	Beta Test results posted (may be sooner)
Fri, April 22, 2011	Final Robots due at 11:59pm
Wed, April 27, 2011	Awards Ceremony and Robot Exhibition

6 Getting Started

6.1 Using battle_v0.p and gsiBot.m

Download battle_v0.p and gsiBot.m from bSpace under Resources and put them in the working directory of Matlab. The function battle_v0.p is what you will use to test your robot. Notice that battle_v0 is a .p file and not an .m file. Files with the .p suffix work exactly like .m files except that they cannot be opened or modified. The reason for locking this function is so that players will not be able to examine its inner workings to find vulnerabilities that can give them an advantage. The function gsiBot.m is a simple robot that you can use to help you understand what needs to be done to make a working robot, and to give you an easy opponent as you develop your own robot.

Later in the project, we will give updated versions of battle_v0, labeled battle_vx, as bugs are found and fixed.

The function battle_v0.p has the following header:

```
[winner, err, errstr] = battle_v0(fun1, fun2, map_name, show, speed)
```

6.1.1 Input and Output arguments

fun1: function handle to player 1 function.

fun2: function handle to player 2 function.

map_name: {'easy', 'sym', 'asym'}: string containing map to play on.

show: {0, 1}: 0 to hide battle, 1 to display it on the screen.

speed: {scalar > 0}: smaller means slower, larger means faster.

winner: {0, 1, 2, []}: 0 for tie, 1 if player 1 wins, 2 if player 2 wins, [] if there is an error.

err: {0, 1, 2}: 0 if neither player errors, 1 if player 1 errors, 2 if player 2 errors.

errstr: struct containing error information, empty if no errors occur. If a player errors, error message can be displayed using the command >>rethrow(errstr).

6.2 Try It Out

To get a feel for how battle_v0.p works, type the following commands into the command prompt:

```
>> [winner, err, errstr] = battle_v0(@gsiBot, @gsiBot, 'easy', 1, 2)
>> [winner, err, errstr] = battle_v0(@gsiBot, @gsiBot, 'easy', 0, 2)
>> [winner, err, errstr] = battle_v0(@gsiBot, @gsiBot, 'sym', 1, 2)
>> [winner, err, errstr] = battle_v0(@gsiBot, @gsiBot, 'asym', 1, 2)
```

Note that the speed input is irrelevant when not showing the game.

6.3 Map Descriptions

In this section, we will describe each map type and how they were intended to be used.

'easy' This map is a basic map containing very few tanks and mines. It is designed to be the first map that you test your robot on. Since this map is extremely basic, it is only intended to test whether your function works (e.g. does it throw an error?) and whether basic skills such as moving towards a goal point and avoiding mines have been programmed in correctly. In the beginning, you may be surprised to see your robot moving in the opposite direction that you wanted it to!

'sym' The symmetric map has more fuel tanks to grab and mines to avoid than the easy map. It is randomly generated so that you can test your robot in a variety of situations. However as the name suggests, this map is symmetric so neither player has a positional advantage. This map is intended to be a primary testing ground for your robots strategy instead of its functionality. You should not move to this map until you are satisfied with its performance for the easy map.

'asym' The asymmetric map has asymetrically placed fuel tanks, mines, and starting positions. Because of this, your robot may have an initial advantage or disadvantage. The purpose of this map is to test how well your robot can capitalize on this advantage or protect itself from an initial disadvantage. Test your function on this map last.

6.4 Reporting Errors and Odd Behavior

If during the course of testing your robot using battle_vx.p you find it behaving strangely. Please let the Head GSI know by emailing him at timmy.siauw@gmail.com. Please have some proof and documentation of the error. For example, one of the robots disappearing from the screen would qualify as odd behavior (that actually happened last time).