# Project I: Moving-target Planner

**Description:**

In this project, you are supposed to write a planner for the robot trying to catch a target. The planner should reside in the robotplanner.m file. Currently, the file contains a greedy planner that always moves the robot in the direction that decreases the distance in between the robot and the target.

The robotplanner function:

```
function[newrobotpos] = robotplanner(envmap, robotpos, targetpos);
```

takes 3 parameters. envmap is the map of obstacles. envmap(x,y) = 0 is free, envmap(x,y) = 1 is obstacle. robotpos is the current position of the robot. targetpos is the current position of the target. The function should return the next position of the robot. It should be any of the 8 cells adjacent to robotpos cell (including the diagonally adjacent cells). It cannot be a cell that is an obstacle or outside of the map boundaries (see current robotplanner for how it tests the validity of the next robot pose).

The robotplanner is supposed to produce the next move within 0.2 seconds. Within that 0.2 seconds, the target also makes one move. But the target can only move in four directions. If the robotplanner takes longer than 1 second to plan, then the target will move by longer distance. In other words, if the planner takes N seconds (rounded up) to plan the next move of the robot, then the target will move by N steps in the meantime.

The directory contains few map files (map1.txt and map3.txt).
Here are few examples of running the tests:

```
>> robotstart = [250 250];
>> targetstart = [400 400];
>> runtest('map3.txt', robotstart, targetstart);

>> robotstart = [700 800];
>> targetstart = [700 1700];
>> runtest('map1.txt', robotstart, targetstart);
```

Executing runtest command multiple times will show that sometimes the robot does catch the target, and sometimes it does not. The letters R and T indicate the current positions of the robot and target respectively. (Sometimes, they may appear as if they are on top of a boundary of an obstacle, but in reality they are not. The letters are just much bigger than the actual discretization of the map.)

NOTE: to grade your homework and to evaluate the performance of your planner, I may use a different and larger maps from the ones in the directory, and a different strategy for how the target moves (different target planner). **The only promise I can make is that the target will only move in four directions and the size of the map will not be larger than 5000 by 5000 cells.**

## To submit:

Submit two files, sent by email to me (maximl@seas):

1. robotplanner.m
2. ASCII file (.txt) with few paragraphs describing the approach you took for the planner (i.e, what algorithm and with what parameters).

## Grading:

The grade will depend on two things:

1. How well-founded and creative the approach is. In other words, can it guarantee completeness (to catch a target), can it provide suboptimality or optimality guarantees on the paths it produces, can it scale to large environments, can it plan within 1 second?

2. How fast (and how consistently) it catches the target