CMRoboBits (15-491) Fall 2007 – Homework 6

Due Wednesday, November 7th 2007 at the beginning of lab (12:30pm)

Introduction "Robotic Tag"

In this homework you are to implement the game of "tag" using the **Scribbler**. You will use the vision system for navigation, similar to the previous homework.

Domain Description

This homework is designed to be competitive. Each team will control one robot and there will be up to three teams (three robots) on the field playing a game of tag. Each robot will be assigned a unique pattern, so you can easily identify and control the particular robot assigned to your team.

The domain is a simple robotic version of the game of tag (also known as "it", or "catch"): one robot is designated the "tagger" which has the goal to "tag" (meaning touch, or in our case drive within a certain distance) one of the other robots which we will call the "taggees". The taggee's role is to run away and avoid being tagged by the tagger respectively. Once a tag occurs, the roles switch and the taggee which was touched becomes the tagger, while the former tagger becomes a taggee.

Scoring: We count positive scores for successfully tagging another robot, as well as negative scores for being tagged.

Important: your program should also be able to run no matter whether there are two or three teams on the field. This is mainly to allow testing in a less cluttered environment and to keep running if one of the three teams fails.

Special Implementation Requirements for "tagging" a robot

Generally, a "tag" occurs when the tagger manages to come within a certain distance of a taggee.

However, since there is noise in your sensory environment and since different teams might query the vision system at slightly different points in time, we need to ensure that both the tagger and taggee-teams agree on when a "tag" happened. The way to implement this is by using different distance thresholds for the tagger and taggee respectively:

When running the taggee (the robot being chased):

your robot should come to a complete stop once the tagger comes closer than **105 pixels** in Euclidean distance, measured between the two centers of the robots, or more formally if:

```
sqrt((tagger_x - taggee_x)^2 + (tagger_y - taggee_y)^2) < 105.
```

It should then **come to complete stop** and **wait for a signal** (dialog box) to confirm to switch to tagger-mode, or whether to resume its old role as a taggee (in case of a false positive).

When running the tagger (the robot chasing the other robots):

Your tagger should continue chasing a taggee until it comes closer than **75 pixels** in Euclidean distance. It should then **come to a complete stop** and **wait for a signal** (dialog box) to confirm switching into taggee-mode, or whether to resume being the tagger (in case of a false positive).

Note, that having two different distance thresholds for the two behaviors should ensure that the taggee will stop once being tagged, and the tagger will continue chasing him a little bit further until it is even closer, thus ensuring that both teams should note tag events. It should go without saying that it is **important that you adhere to the above rules to ensure a fair game of tag.** Obviously, if the tagger will stop, this means that the taggee should definitely have stopped already due to its wider tagging threshold. In order for this to work, you will need to query the vision system frequently enough, in other words, keep your robot's step-size small enough, or run the vision queries concurrently using timers and variables, rather than a single closed loop. Also, please have your robot stay within the constraints of the visible region (don't drive of the field to avoid being tagged).

Strategy ideas

Tagger Strategy Considerations

You might want to come up with a good heuristic on which robot to chase. This could be based on e.g. the shortest distance to the target. It is however important to avoid oscillations. You might want to consider a finite state machine approach where you stick with a robot for given amount of time. Or you might want to consider using hysteresis to switch your target only if the new target is significantly closer than the previous one.

Taggee Strategy Considerations

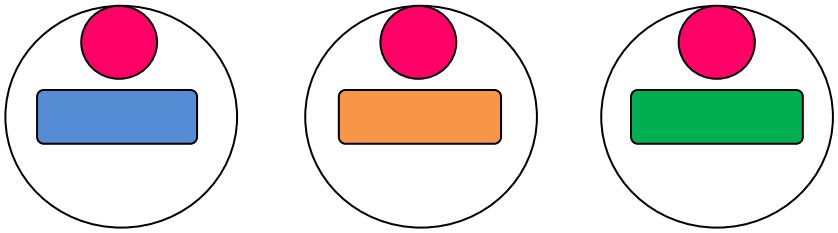
Try to be creative with your avoidance strategy. You might want to consider sampling points in space and evaluating them based on their distance to the tagger and/or other taggees.
You can also try more elaborate strategies (such as hiding behind the other taggee). If you use multiple behaviors, you might again want to combine them in a state-machine.

Vision Patterns

The robots (and thus the teams) on the field will be identified by their center blob which will be either "blue", "orange", or "green". To obtain your orientation, a pink dot is placed at the top of your robot.
Note the new color IDs when querying the vision service:
4 = blue
5 = orange
6 = pink
7 = green

Please only use these color-IDs since they are well-calibrated and well-separated in the color-space. **Any other colors (yellow, white, red, etc…) are no longer clearly calibrated and you should refrain from using them for this homework and your project.**

The following image demonstrates the new robot-patterns:



Flow of your program:

You should be able to run any of the three robot colors (and track the other two). Your program should allow an easy setup of which robot to control and whether this robot should initially carry the role of the tagger or taggee. **Please use dialog boxes to allow changing these settings at runtime.** Also, make sure to use dialogs when switching roles as mentioned previously.