Programming Assignment: Multi-Robot Formations

In this assignment, you will need to use the handout from last week on how to implement multiple robots in the Nomad 200 Simulator. This handout is posted on our course web page and is available online at:
http://www.cs.utk.edu/~parker/Courses/CS594-fall02/Assignments/Nomad200-multirobot.pdf. Code for helping you with the multi-robot communication is also posted on our course web page and is available online at:
http://www.cs.utk.edu/~parker/Courses/CS594-fall02/Assignments/communicate.tar.

For this assignment, you will generate code that allows 4 robots to move in side-by-side formation from one waypoint (i.e., x,y location) to another. The robots should maintain a perpendicular distance \( d \) between them. Only one of the robots (the “Leader” robot) knows the waypoints to be followed. The rest of the robots must move in order to stay in formation with a specific reference neighbor. The following diagram illustrates this objective:

In this diagram (and for what you will implement), Robot B is the leader robot, and is the only robot that knows the location of the waypoints. The Robot B behavior for moving from waypoint to waypoint is the same as your go-to-subgoal behavior that you have programmed previously.

Robots A, C, and D must make movements based upon the current location and heading of their own reference neighbor robot. In the diagram above (and for the code you implement), the assignment of reference robots is as follows:

<table>
<thead>
<tr>
<th>Robot</th>
<th>Reference Robot</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>C</td>
<td>B</td>
</tr>
<tr>
<td>D</td>
<td>C</td>
</tr>
</tbody>
</table>
In other words, Robots A and C move based upon the current position of Robot B, while Robot D moves based upon the current position and heading of Robot C. Rather than trying to sense this position and heading information using the sensors, we will use communication to simplify the approach. Thus, robots must communicate with each other to share their own current position and heading information with their teammates, so that this information can be used for velocity and steering control in robots A, C, and D. You can either use broadcast or point-to-point communication to accomplish this. Just ensure that a robot only uses position and heading information from its reference robot to make its own control commands. When the follower robots A, C, and D receive the current position and heading information from their reference robot, they calculate their own desired position in the formation as the location a distance $d$ from the reference robot, perpendicular to the heading of that reference robot.

To simplify this assignment, we make the following assumptions/constraints:

- There are no obstacles, so robots do not need behaviors to avoid obstacles.
- Robots should never move backwards. If a robot is in front of its desired position (as determined by examining the current position and heading of its reference robot), then it sits still (and perhaps rotates in place) until the desired position is in front of the robot.
- Your code does not have to handle turns of more than 45°. Turns of much more than 45° would mean that neighboring robots are blocking the path of the leader at the point of the turn, which greatly complicates the formation control.
- The leader robot does not attempt to monitor the progress of the follower robots. That is, Robot B will just move through the series of waypoints provided to it and will stop when the final waypoint is reached without paying any attention to the other robots (other than to communicate its own position and heading information).
- Because of the above constraints, it is likely that your robot team will not remain in perfect formation throughout the task. More than likely, the leader Robot B will be slightly ahead of Robots A and C, and Robot D will be slightly behind Robot D. To help minimize this, you should make your velocity control for Robots A, C, and D a function of the distance between the robot’s current position and its desired position. The farther away the desired position is, the faster it moves. This should help the follower robots keep up with the leader.
- Robots are in the proper formation at the beginning of the task, facing in the direction of their next waypoint.

For this assignment, use the following waypoints and parameter settings:

<table>
<thead>
<tr>
<th>Waypoint/Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Starting position</td>
<td>(x,y) = (0,0)</td>
</tr>
<tr>
<td>Waypoint #1</td>
<td>(x,y) = (2000,0)</td>
</tr>
<tr>
<td>Waypoint #2</td>
<td>(x,y) = (5000,1400)</td>
</tr>
<tr>
<td>Waypoint #3</td>
<td>(x,y) = (8000,1400)</td>
</tr>
<tr>
<td>Inter-robot distance, $d$</td>
<td>500</td>
</tr>
</tbody>
</table>

You should strive to make the motions of all the robots as smooth as possible.

**EXTRA CREDIT** (worth 10 points):

For extra credit, create a metric that measures the instantaneous error in the formation at any given point in time during the task. Measure the error every $\Delta t$ simulation seconds (on the order of every 5-10 seconds or so) and plot the instantaneous formation error at each time step from the start of the simulation to the end. This error should be measured automatically and electronically by one or more of your robots (or a separate process) while the robots are moving along the path. Do not measure the error manually.
TURN IN THE FOLLOWING:

a) A hard copy of your complete code, fully documented.

b) Seven (7) screen dumps of your code during operation, which show the world view of the robots in the situations given below. (By “world view”, we mean the window in which all 4 robots appear together.) In each of these screen dumps, mark (either electronically or manually) in the world view the location of the starting point and all waypoints to serve as reference positions.

The seven situations for your screen dumps are as follows:

1. At the beginning of the task, with the robots in the correct starting formation.
2. When the leader is half-way from the starting point to waypoint 1
3. When the leader is at waypoint 1
4. When the leader is half-way between waypoints 1 and 2
5. When the leader is at waypoint 2
6. When the leader is half-way between waypoints 2 and 3
7. When the leader reaches waypoint 3

c) A discussion of the following topic: What do you think would be the best way to handle the formation control if the path to be followed had to make sharp (say, 90°) turns? Can it be accomplished using only local information regarding the reference robot position and heading? If so, how? If not, why not?

d) A discussion of the following topic: What do you think would be the best way to handle the formation control if the group formation moved through obstacles along its path? (That is, the leader robot follows the path without encountering obstacles, but one of the other robots encounters obstacles as it is attempting to remain in formation.)

e) EXTRA CREDIT:
   - Formulate and explain your metric for calculating instantaneous formation error as a mathematical function. The notation should be precise and explicit.
   - Create a plot of the instantaneous formation error from the beginning of the task until the last waypoint is reached. Annotate your plot with notations to indicate where the 7 situations from part b) above occur.

f) Email an electronic version of your code to parker@cs.utk.edu, naming your code yourlastname-5.c (or .zip, etc.)