CS 69.13, Fall 2022, Elliot Potter

Method Description

There are two major components to my implementation: 1. Environment construction. This is done in the launch files and world files. I constructed `wall.world` (for Gazebo) from scratch; I copied the stage worlds from `stage_ros` and used GIMP to rewrite the PGM file to something simpler than willow-full. In terms of my launch files, I'm particularly proud of `flocking_stage.launch`, because I was not fully sure that `launch` files worked with stage. 2. Code: The node `flocker` contains the driver code, `main` function, and all stateful code, as well as all callbacks. The module `boid.py` contains the code for calculating the desired direction of the robot. This is the section that actually deals with flocking. A note – I changed the separation code to exert force based on 1/distance_between_bots, rather than based on distance_between_bots (as I think you had in your slides? Maybe I misremember?). This makes sense because force should be more intense as the boids get closer together. The module `wall_avoider_2.py` calculates the force vector away from walls, as detected by the laser scanner.

```
I chose to have my robots drive continuously and change direction fluidly, in the same way t
It would probably be easier to debug the system if the robots planned and executed discrete
went against the spirit of the assignment.
```

Evaluation

The WINS

1. Cohesion. It works in both environments!
2. Avoiding walls in Gazebo, or with a single robot in Stage
3. Avoiding each other in Gazebo
4. Figuring out all the more annoying environment controls; for instance, different scan topics in Gazebo and Stage, creating worldfiles and maps for Stage, slightly different odom message structures, how to create robots in each case.

The LOSSES

There are a few major issues with my implementation;

1. The robots collide with walls (in Stage), given enough time in the simulator. In Stage, this is the result of blind spots on the robot – once it rotates away from the wall, it can no longer see the wall, and it rotates back to face the wall. For a single robot, I patched this by including a weighted

average of historical force vector away from the wall. You can see this with `roslaunch flocker collision_stage.launch`. For the most part, it looks like multiple robots will avoid walls in Gazebo, but drive straight into them in Stage.

2. Flocking behavior is not super apparent in either Stage or Gazebo, because they appear to require different hyperparameters. In Gazebo, the robots spend a lot of time smoothly drifting in circles, while in Stage, they seem to prefer driving in figure-8s. Similarly, I had to raise the wall repulsion a lot in Stage to try to get the bots to crash into the walls less often; instead, they got overwhelmed and crashed into one another. In Gazebo, the bots basically don't collide, but the higher repulsion that I needed for Stage means the bots spread out a lot more.

3. The robots are far more interested in driving in weird patterns than demonstrating alignment. This would probably change if I boosted the alignment hyperparameter, but the robots would just hit more walls.