Plowboy

Jonathan Bluhm jobl6075@colorado.edu
Sebastian Brunet- sebr8260@colorado.edu
Glenn Holzhauer - glho1292@colorado.edu

Abstract

Our initial plan was to develop a snow plow robot car, and we did end up developing a car that could plow decently well. However, we did not feel like a car could perform some of the driving patterns that we wanted. So after some discussion, we decided to go with an e puck as our robot, and go for a proof of concept instead. Our initial plow will spin in circles. If the plow sees snow, it will drive towards it and push the snow out of the boundary. Once sufficiently close to the boundary, the e puck will reverse and go back to the center and continue rotation. Only a camera will be used to sense the world. The goal of this is to see if we could get a plow working with just the camera. This way these plows could be made cheaply in the real world; parking lots would always be clear, allowing plows to spend more time plowing roads.

Equipment

We used the e-puck robot fixed with a camera, a light (used to amplify the colors) and a gps. We decided on not using the car because the e-puck was easier to work with with regards to the added features we needed and writing a controller. We used images from the camera to detect snow. We “chop up” the image into three chunks from left to right. This allows us to know whether snow is on the left right, or in the middle. While a distance sensor would have made things easier for us at some points, we decided not to use one and instead use only color detection as a way to know how far away from things the e puck is.

Deliverables and Implementation Plan

1. World import and creation - Deadline (4/10), Lead (Glenn)

We decided to go with an existing world from WeBots. We chose a square world and created a boundary that was colored bright green so that the robot could distinguish a boundary line to which it would push snow objects. Our snow objects are essentially white colored balls that the robot can push around using its plow. We began with a combination of Blender and OpenStreetMaps importing to create a large scale environment from Boulder, but ultimately found that WeBots had serious performance issues with these maps, and testing our project would have been extremely challenging. Testing various robots on a larger scale, we found they were challenging to properly configure for the task at hand, and a small world was more fitting.

2. Plow creation - Dead (4/12), Lead (Glenn)

After testing many robots on wheels, we eventually realized that the best robot equipped to attach a reasonable plow to and navigate a space simply was the e-puck. We created a plow object to fix on our e-puck to be able to pick up several snow objects at a time. The way we created the plow was by adding three boxes to the turret slot on the robot with translation and size configuration to form a U shape. To avoid other physics and collision issues, we applied a very low mass to these boxes.

3. Mapping of snow - Deadline (4/18) Lead (Johnny)

We did not end up using mapping to determine the location of snow objects. Initially we wanted our robot to behave as a sort of “roomba” in the sense that it would identify the location of snow objects by searching the locations that it had not yet visited. We decided that this might be too similar to lab 5 and opted for a real-time snow detection technique instead. This technique relies on the use of the camera in combination with a flashlight (to amplify colors) and essentially searches for white pixels in the robot's view. We set a threshold for the number of pixels or intensity of white that appeared in the camera image and used that to determine snow objects. All the values used for our threshold were obtained and tuned experimentally by moving the e-puck to different distances from the snow objects and recording the white pixel count.

4. Obstacle evasion - Deadline (4/24) Lead (Sebastian)

This did not end up happening either. Instead our challenge was to be able to detect when we are close to the wall, and then backup and reset position. We did this by using green detection with our camera. We made our wall bright green and then looked for green pixel values. Once the observed green values exceed a certain threshold, we know that the e-puck is close to the wall and can begin to reverse.

5. Pathing for missed snow - Deadline (4/25), Lead (Sebastian)

We designed our map to contain snow objects in the form of white colored balls. We determine how long the e-puck should push the snow objects before reversing (in order to get them to the boundary line) and how long to reverse for by tracking the number of timesteps that occur while pushing using a loop counter. Once the e-puck reaches the wall, we reverse until that loop counter reaches 0. The robot then spins freely trying to detect snow objects and once it does it repeats the process.

6. Dynamic thresholds and auto stop - Deadline (4/27), Lead (Johnny)

The epuck can sometimes get stuck in an infinite loop approaching snow that is already plowed. This occurs because of the changes in colors caused by the shading. So the robot thinks the snow is closer than it actually is. We solved this issue by keeping track of the angle that the plow is on. If it had previously used that angle, then we “skip” plowing that snowball. A simple fix. Another issue is that as the e puck plows snow, the snow clumps up in areas. This leads to false positives occurring (the robot thinks it needs to plow the snow, but it is already plowed). To solve this we created dynamic thresholds. If the robot “free spins” (not plowing) for 1 second we increase our white threshold by 2. This leads to the e puck still detecting nearby patches of snow, while ignoring the farther away ones that it would have detected had we not raised the threshold.

Final Submission Video

The e-puck will be shown plowing a square map meant to simulate a parking lot, by using color detection technique to plow the lot all on it’s own. The snow will be represented by white spherical objects.