Lab 4: iRobot Navigation in C

Sam Mansfield and Toan Vuong
TA: Hoekun Kim
EECS149

October 2nd, 2013

1 Introduction

In this lab, we improved the iRobot's navigation system. By first designing an algorithm, then implementing it using state machines, we allowed the iRobot to get around obstacles to continue on its original direction.

2 Analysis

Our algorithm is fairly simple. Our robot will start off moving forward. When it detects either an obstacle or a cliff, it will move backwards, turn right, then attempt to turn left to return to its original trajectory. If this fails because another obstacle is in the way, it will attempt the algorithm all over again. The figure below demonstrates this.

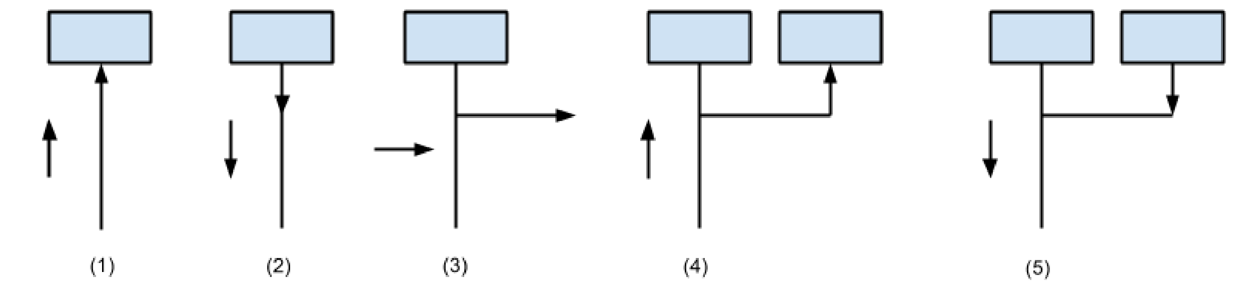

Figure 1: Algorithm Part 1. Single obstacles. Arrows denote direction of travel.

It works on the basis that, in order to return to the original direction, our robot must turn left once for every right turn it makes. This leaves the robot to move forward in its original trajectory. in its original orientation This means that we have to keep track of the number of obstacles that the robot has encoutered. Everytime we successfully dodge an obstacle, this obstacle counter decreases by one.
In Figure 2, our robot is in the process of avoiding an obstacle when it reaches another obstacle. In state (7), its obstacle count increases to 2. It begins the obstacle avoidance algorithm in (8), and (9). Figure 3 shows state (10), where the robot has successfully made two right turns, corresponding to two increments of the obstacle counter, and one left turn, corresponding to one decrement of the obstacle counter.
Thus, in (11), when the robot makes the final left turn, the number of obstacles decreases to zero and it is back facing its original direction again.

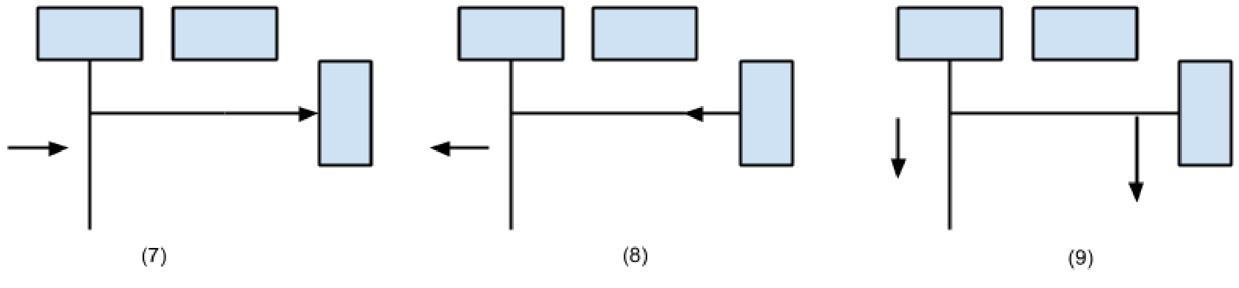

Figure 2: Algorithm Part 2. Encoutering an obstacle while avoiding the first. Arrows denote direction of travel.

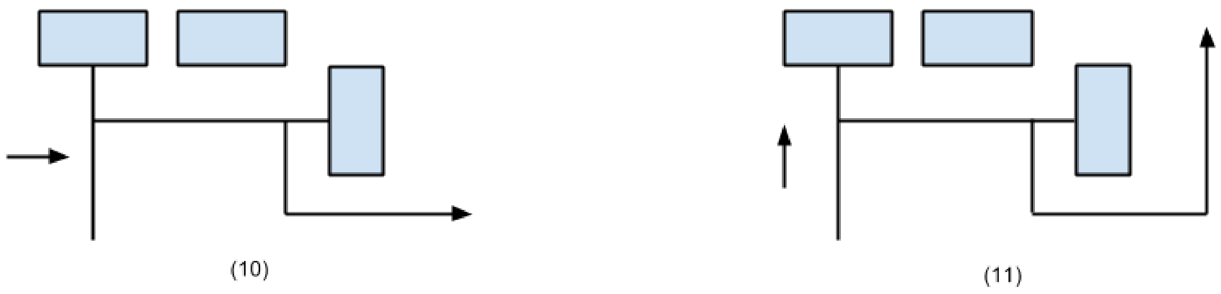

Figure 3: Algorithm Part 3. Recovery from both obstacles. Arrows denote direction of travel..

To accomodate this algorithm in the robot's states, we ended up adding two new states, BACK and TURNRIGHT. The BACK state goes backwards for 50 mm, the TURNRIGHT state makes a right turn. We reused the existing TURN state as a TURNLEFT. Figure 4 displays our final FSM.

At the beginning of the lab we tried to create a more elaborate FSM and realized that we were actually making things more complicated to debug and therefore implement. By simplifying our design we created a better FSM. It was a little unclear if we were supposed to use the square pattern that initially boots up or to just make the robot go straight. Initially we created an obstacle avoidance algorithm that only worked for the square pattern. It was only later on that we realized that we were supposed to create an obstacle avoidance algorithm for a straight line.

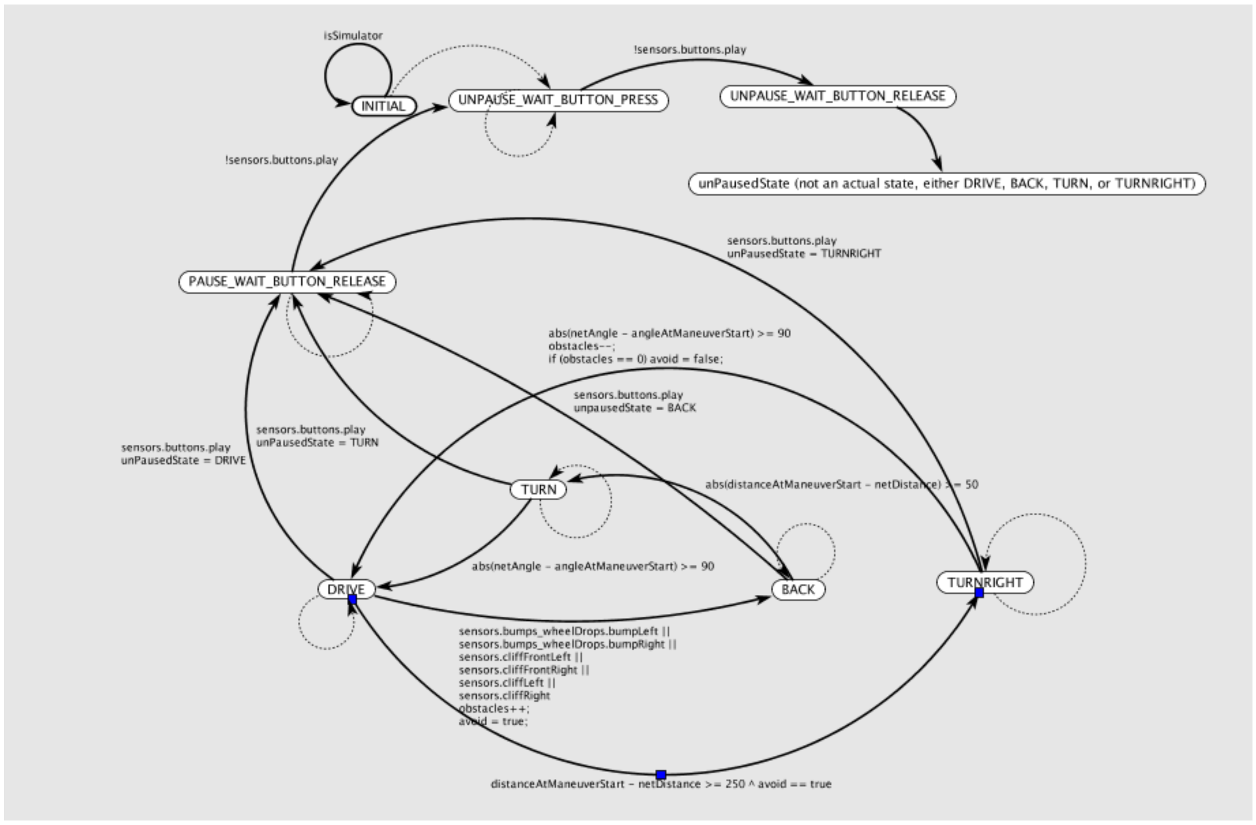
isSimulator
INITIAL
UNPAUSE_WAIT_BUTTON_PRESS
!sensors.buttons.play
UNPAUSE_WAIT_BUTTON_RELEASE
unPausedState (not an actual state, either DRIVE, BACK, TURN, or TURNRIGHT)
!sensors.buttons.play
sensors.buttons.play
unPausedState = TURNRIGHT
PAUSE_WAIT_BUTTON_RELEASE
abs(netAngle - angleAtManeuverStart) >= 90
obstacles--;
if (obstacles == 0) avoid = false;
sensors.buttons.play
unpausedState = BACK
sensors.buttons.play
unPausedState = TURN
sensors.buttons.play
unPausedState = DRIVE
TURN
abs(distanceAtManeuverStart - netDistance) >= 50
abs(netAngle - angleAtManeuverStart) >= 90
DRIVE
BACK
TURNRIGHT
sensors.bumps_wheelDrops.bumpLeft ||
sensors.bumps_wheelDrops.bumpRight ||
sensors.cliffFrontLeft ||
sensors.cliffFrontRight ||
sensors.cliffLeft ||
sensors.cliffRight
obstacles++;
avoid = true;
distanceAtManeuverStart - netDistance >= 250 ^ avoid == true

Figure 4: Obstacle/Cliff avoidance FSM implemented in Lab 4

3 Conclusion

Designing an obstacle/cliff avoidance algorithm was quite fun and rewarding when it actually worked. This was a very applicable problem for state machines and we can apply this technique to future problems that are similar in nature. This lab also helped us think about solving very simple human problems (avoiding an obstacle) step by step so that it can be applied to a robot.