Emily Wang
09/19/2014
Computational Robotics: Warmup Project

Personal Learning Goals for the Warmup Project

- Become familiar with ROS tools
    - resulted in tradeoff of time devoted to project: spending time on tutorials/poking around vs. implementing something complicated with the behaviors)
- Become familiar with neato platform (lidar and motor capabilities, visualizing results in simulator or printing msg content, etc)
- Introduction to the robotics challenge mindset - working with sensor measurements and FSM control, and debugging!

Wall Following Strategy

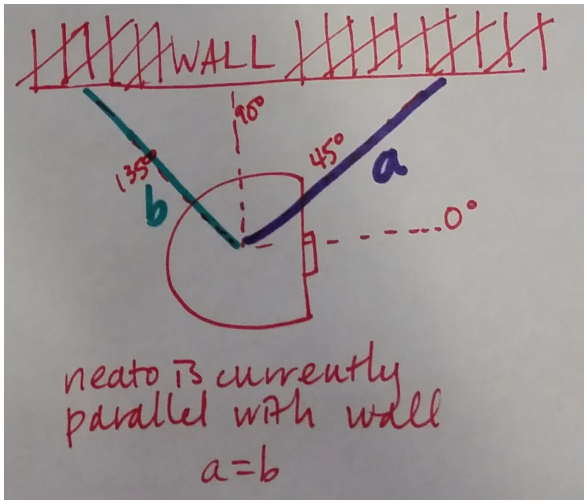
Assumptions:

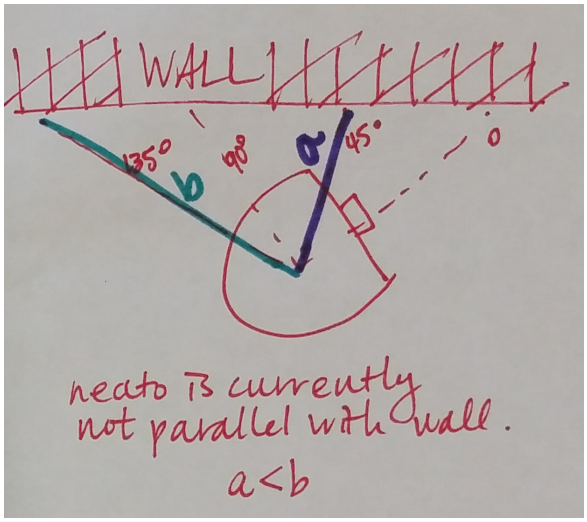
- Wall is on left-hand-side of robot
- Wall is within lidar range (0m<distance<5m)
- No corners to turn
- No obstacles in robot's environment

The Plan:

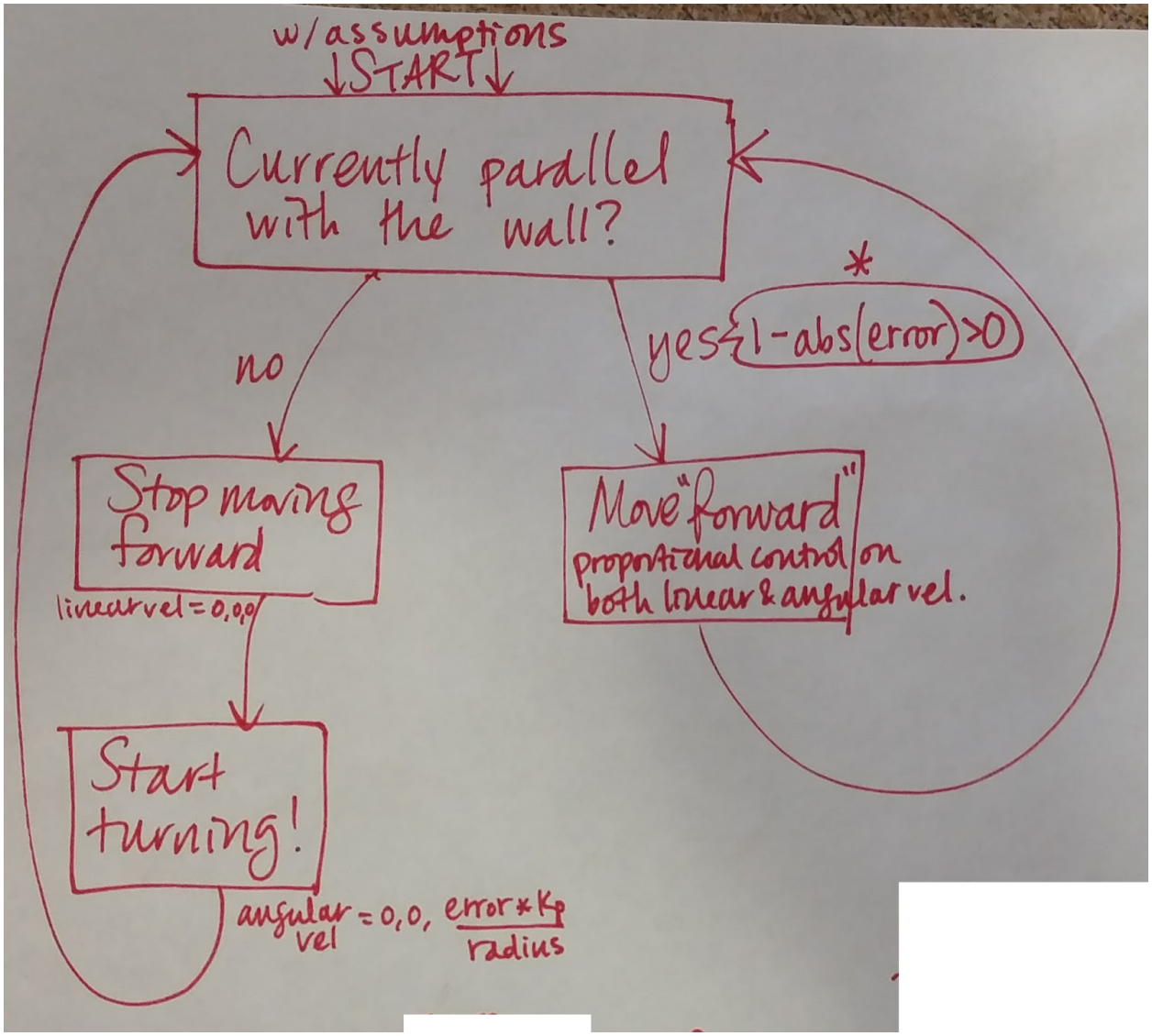
- Measure distance from neato to wall at 45 degrees (distance a) and 135 degrees (distance b)
- Turn/Move neato the necessary amount to make a=b so the robot will be parallel to the wall
- Move forward and do small adjustments as necessary to stay parallel

Doodles for showing the parallel/not parallel situations:





Finite State Machine Control Diagram:



Wall Follower Code Architecture

(relevant script: comprobo2014/src/warmup_project/scripts/wall_follower.py)

class Wall_Follower:

> """ Uses proportional control to travel in a path parallel to the wall on the left-hand side of the neato.
>
> Currently assumes that the wall will be on the left-hand-side of the neato and that there are no corners to turn while following."""

(continued on next page)

```
        def __init__(self):
        """
        Sets up an instance of a Wall_Follower() object.
        Things that happen when a new instance is made:
        - Set up the 'wall_follower' node
        - Set up a 'cmd_vel' publisher
        - Set up a 'scan' subscriber, with the process_laser_scan() method being the callback
        - Initialize variables for lidar measurements and proportional control
        (to be used in other Wall_Follower methods)
        """

        def process_laser_scan(self, msg):

        def run(self):
        """
        Method for executing the wall following behavior.
        Sends Twist messages to the neato depending on the
        state of the system (i.e. looks at what self.error's
        current value is) while measurement analysis/computation
        happens in self.process_laser_scan.
        """

if __name__ == '__main__' :
        try:
                controller = Wall_Follower()
                controller.run()
        except rospy.ROSInterruptException: pass
```

So… does it work?

When tested in an environment that agrees with the given assumptions (wall is on left-hand-side, no corners to turn around), the robot works well. The effects of the proportional control are apparent - the robot does dramatic changes at first and slows down as it approaches the "target" (i.e. being parallel with the wall). The robot was only tested in the AC128 hallway area; so the proportional constant (self.Kp) may or may not need to be tweaked.

Admittedly, this wall follower code is a naive implementation that has the potential to become more intricate/elegant to handle a variety of scenarios and environments. A lot of my time was spent getting familiar with ROS, attempting to reconsider the wall following strategy with different functions/finite states, and much debugging… (My estimate is that I spent around 12

hours on this project.) Even though I wasn't able to make a very complicated wall follower this time around, I now have a list of "things to do in future ROS projects!!" that I will be sure to remember and do for future robotics challenges. Overall, I had a lot of fun with this warmup (my first dive into the world of robotics as an engineering student) and am very much looking forward to the rest of the semester!

If I had more time for the warmup project, what would I try next?

- Wall follower functionality for both the left-hand and right-hand side and turning around wall corners
    - Taking data from both the 90 degree region and 270 degree region… (So, taking data from two different regions, and telling the neato to pick the closest/most feasible wall to follow).
- Obstacle avoidance or people follower:
    - Since these behaviors involve some sort of object tracking, applying meanshift on the lidar data for determining the center of an obstacle to avoid/person to follow could be effective…
- Finite state machine control for different behaviors
    - How will the neato determine whether it should be following a wall or avoiding a person? Does the algorithm need to consider the cases differently or is it better to hybridize the two different behaviors? Are additional sensors needed for the robot to make a better decision on what to do next?

Insights/Reflections

(good things to do with comprobo projects; didn't necessarily do all of these in the warmup project but will remember them for all assignments/projects from now onwards)

- Object oriented programming. write classes for ROS nodes
- Finite state machine for figuring out and communicating decisions/thought process!
- Draw lots of pictures! geometry diagrams are very very helpful
- Start with naive implementation and list of assumptions, then make it more robust
    - Once you get to something testable in the real world, the real world results will make it very obvious what needs to be fixed/worked on next.
    - As you tackle each assumption, the implementation become more robust and work better in different situations.