Homework 3: Laserfollow

Our group unfortunately was unable to fully complete Laserfollow.

Let's take a look at our laser_callback function.

```
def laser_callback(self, las):
    ranges = las.ranges
    left = len(ranges)//4
    right = (len(ranges))//4*3
    back = len(ranges)//2
    left_front = len(ranges)//8
    right_front = (len(ranges))//8*7
    front = -1
```

We began our wall following logic by finding the LIDAR range indices for 4 orienting directions.

```
    if not self.following:
        if ranges[front] < .5:
            self.following = True
        else:
            forward = Twist()
            forward.linear.x = .1
            self.publisher.publish(forward)
```

Then we checked if *qbert* was actively tracking a wall. While *qbert* was not following a wall, we would publish a forward linear `Twist`. Our threshhold for detecting a wall was `.5` meters.

```
 else:
    twist = Twist()
    if ranges[front] < 0.4:
        twist.linear.x = .1
        twist.angular.z = 0.05
        self.publisher.publish(twist)
```

Whenever *qbert* detects a wall, we publish turning `Twist`s until the wall is no longer detected directly in front of *qbert*. This bare-bones logic was not sufficient enough for *qbert* to succesfully turn away from forward obstacles.

```
else:
    twist.linear.x = .1
    right_average = (ranges[right]+ranges[right+2]+ranges[right-2]+ranges[right-3] +ranges[right+3])/5
    if right_average <= 0.4:
        twist.angular.z = 1.25 * (0.4-right_average)
    self.publisher.publish(twist)
    left_average = (ranges[left]+ranges[left+2]+ranges[left-2]+ranges[left-3] +ranges[left+3])/5
    if left_average <= 0.4:
        twist.angular.z = -1.25 * (0.4-left_average)
    print(f"front: {ranges[-1]}, left: {ranges[left]}, right: {right_average}")
    print(f"{self.following}")
if ranges[front] > 0.4:
    twist.angular.z = 0.0
    self.publisher.publish(twist)
```

Once *qbert* successfully turns away from a wall, we maintain a forward velocity of `.1` m/s. We take the average of several LIDAR readings from the left and right of *qbert*, and make decisions to veer right or left depending on those readings. Our final attempt at solving what magnitudes of turning we would publish involved subtracting the average reading of a direction from a constant, then multiplying by a constant scalar. This proved unsuccessful.

Furthermore, the last `if` statement in the above code snippet retrospectively makes no sense. We think it was a remnant of a past attempt at solving Laserfollow.

Review

Given more time, we would want to probably use more samples in our `left_average`, `right_average` variables. We would also want to further fine tune our magnitude solution.

Retrospective

Start:

- Doing more remote work with bag files

Stop:

- Missing class