CSE 397/498-013
Introduction to Mobile Robotics

Homework: Lines, Walls & RANSAC
Report Due Date: Thursday, 20 Oct 05 submitted via Blackboard
PRIOR TO THE START OF CLASS (**NO ANALOG SUBMISSIONS**)

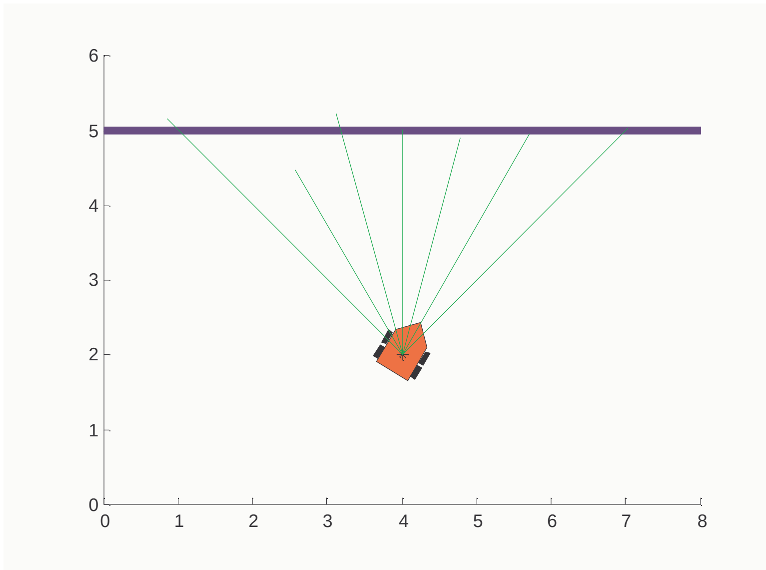


Figure 1: Corrupted Sonar Range Measurements

A. Objectives:

1. Practice with Jacobians and uncertainty propagation
2. Generation of sensor models.
3. Least-squares approaches to fitting lines.
4. Investigate the use of RANSAC for robust line fitting techniques

B. Robot Model:

1. We are using sonars to estimate our distance to the wall.

C. Requirements:

1. You MUST use Matlab to complete this assignment.
2. This is an individual assignment. Each student is required to submit his/her own work in order to receive credit.
3. As stated on the web page, if you turn this assignment in late without coordinating with me first you will receive a 0.

D. The Assignment: Remember that

- **no work $\Rightarrow$ no points**
- **imaginary steps $\Rightarrow$ imaginary points**

1. **Jacobian Exercise:** Calculate the Jacobians for the following vector functions.

    a. $$\begin{bmatrix} x \\ y \\ \theta \end{bmatrix} = \begin{bmatrix} r\cos\theta \\ r\sin\theta \\ \theta \end{bmatrix}$$

    where you are measuring *r* and θ and need to estimate you pose $[x, y, \theta]^T$

    b. $$\begin{bmatrix} x \\ z \end{bmatrix} = \begin{bmatrix} \frac{bu}{f\cot\alpha - u} \\ \frac{bf}{f\cot\alpha - u} \end{bmatrix}$$

    where you are measuring *u* and need to estimate the state $[x,z]^T$.

    c. For problem D.1.b above (which is from our structured lighting example in case you've already forgotten), assume that the measurement *u* has a variance of σ_u^2. Propagate this into uncertainties in our position estimates for *x* and *z*.

2. **Least Squares Exercise:**
    a. For this problem, we will be using the *weighted* least-squares equation Equation 4.73-4.74 in the text
    b. Implement a function `[r,a] = WLS_Line(rho,alpha,w)` in Matlab that calculates the weighted least-squares fit of the line. The function takes an ***array*** of range measurements `rho` with a corresponding array of bearing angles `alpha` and weights `w`. It returns the line parameters [r,a] that corresponding to range and normal angle.
    c. Note you ***must*** use the `atan2` version of the function.
    d. TIP 1: Both the numerator and denominator should be multiplied by -1 for the `atan2` function to work correctly.
    e. TIP 2: You may get a negative radius when doing this exercise. This is not wrong.

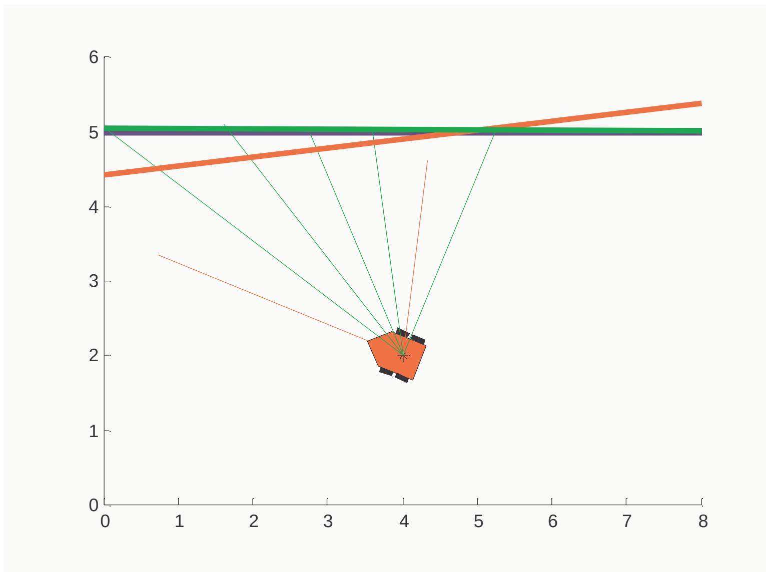
3. **Sonar Sensor Model:**
    a. In this problem, you will generate a sensor model for sonar measurements.
    b. Assume that your sonar has a maximum range of 5 meters.
    c. Assume that the range measurements are corrupted with random zero-mean Gaussian noise with standard deviation = $0.02\rho^2$, where ρ is the actual range to the target.

d. Further assume that 20% of the sensor measurements suffer from multi-path, resulting in a range measurement error of +20-50%. You may assume that these measurements are uniformly distributed.
e. You are to implement a function `rHat = FireSonar(r)` that:
   i. Takes an ARRAY of true ranges to a target as an input.
   ii. Tests to see if each range was affected by multi-path error. If so, the appropriate random error is added to the actual range measurement and returned from the sensor model as `rHat`.
   iii. Otherwise, the range measurement is corrupted with appropriate random Gaussian noise and returned as `rHat`.
   iv. Under no circumstances should the range returned exceed the maximum detection range of the sensor. For ranges >5 m, the sonar will return a value of -1 (indicating that no target was detected).
f. Helpful functions:
   i. `rand`
   ii. `randn`

4. **Wall Fitting:**
   a. Assume the existence of a wall at y=5 meters as in Figure 1 above.
   b. Your robot is equipped with a ring of 25 sonars spaced in 15 degree increments [pi/12:pi/12:2*pi].
   c. Your main function hwk3 will:
      i. Take as mouse input the robot position with some random orientation.
      ii. Determine the range from each sonar to the wall and pass the appropriate range and bearing arrays to `FireSonar`
      iii. From the corrupted range measurement, you will form an estimate of the position and orientation for the wall relative to the robot. To do this you will employ your `WLS_Line` function from part 2 above.
         1. You must choose your weight appropriately (refer to the lecture notes or Chapter 4 for a good weight metric).
         2. Your MUST NOT pass invalid range measurements `WLS_Line` function!
      iv. Redraw the new wall as shown in Figure 2 below (red wall).
   d. Repeat c.ii-iv above employing RANSAC to filter out outliers from your sonar measurements.
      i. Determine the maximum number of sample trials `n_max` that you need to run RANSAC to ensure you have a 99.9% probability of choosing the correct line (you may assume a 20% outlier rate, although this is technically not quite correct).
      ii. Use the subset of sonar measurements that yields the largest consensus set after `n_max` trials to fit the wall.

iii. Also redraw the new wall as shown in Figure 2 below (green wall).

e. In your figure, sonar measurements that were used in part d above should be drawn green, while outliers are drawn red.



G. Additional Questions:

a. If you have 10 range measurements with each having a 50% chance of being an outlier, how many random sample trials would you have to run to have a 99% probability of successfully fitting the correct line.

b. What about 99.9%?

c. TIP: Look at the CVOnline link from the course web page.

H. Turn in: A write-up, to include images from all simulation trials, answers to all questions, as well as your Matlab source code.