Peter Huynh
November 1, 2016
CSCI 3302
Lab 4

Localization

Goal:

- Implement a simple feature detector using the ultrasound receiver
- Use range and bearing information to features in the environment to improve the robot's position estimates
- Get practical experience with error propagation

Required:

- A Sparki robot
- At least two PVC pipes of approximately 2 inch diameter and higher than 12cm ("beacons")

Overview:

Preventing build-up of a robot's position estimate requires additional information, such as range and/or bearing to known features in the environment. In this exercise, we will use Sparki's ultrasound ranger to identify a set of beacons (PVC tubes) in the environment and calculating their relative range and bearing to Sparki. Fusing this information into an improved position estimate requires the following steps:

1. Action update: Calculating a position estimate of the robot using odometry and the error propagation law.
2. Perception update: Calculating a position estimate of the robot using the known location and measurements of range/bearing to 2 or 3 beacons.
3. Estimating the variance of the so-obtained pose using the variance of the ultrasound ranger and the error propagation law.
4. Calculating the new pose as a weighted sum of the robot's previous estimate and the pose provided by the beacons using the inverse of its invariances as weights. (High variances will lead to this estimate becoming less influential.)

This lab will focus on step (1) and (3).

Instructions:

1. Implement a detector for the PVC pipes. Make sure the environment is clear of clutter so that the ultrasound sensor only sees the upright PVC pipes on the table. Use the following code to get started. Use #define NO_LCD to display the LCD and free up a lot of Sparki's memory:

```
#define NO_LCD
#include  <Sparki.h>; // include the sparki library
int angle=-30;
void setup() { }
void loop() {
 sparki.servo(angle);
 angle=angle+1;
 if(angle>30){
   angle=-30; Serial.println();
 }
int cm = sparki.ping(); // measures the distance with Sparki's eyes
Serial.print(cm);
Serial.print(" ");
}
```

You can now inspect the ultrasound readings in the serial terminal. For example, sweeping from -30 to 30 degrees in 1 degree intervals yields the following readings:

First measurement: 72 64 39 39 39 39 40 40 39 39 39 38 19 19 19 18 18 18 18 18 18 18 18 18 18 17 18 18 18 18 18 18 18 19 18 19 56 57 57 62 63 62 62 62 62 62 61 61 61 61 61 61 62 63 64 69 69 70 70 72 71
Second measurement: 64 46 48 49 49 49 48 48 48 48 47 47 47 47 47 49 48 48 14 12 12 13 12 11 12 11 12 11 11 11 11 11 11 11 11 10 11 11 11 11 11 10 10 10 12 11 11 11 11 11 12 12 11 12 12 52 53 65 66 65 64
Third measurement: -1 4 48 48 48 48 48 48 48 48 48 49 49 49 6 6 5 5 4 4 4 4 4 4 4 4 3 4 4 4 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 4 3 3 3 4 4 4 4 5 4 5 5

Corresponding to measurements at 18cm, 11cm and 3cm, respectively. Notice that the angle at which the pipe is discernible is quite large. It makes sense to store these values in an array and then look for values that are feasible, e.g. by establishing a maximum distance that the beacon can be at. (The data shown here contain some clutter on the table.) One way of doing this is to look for changes that are larger than 15, e.g., to detect the presence of an object.

2. Experimentally estimate the variance of the ultrasound range sensor (σ^2_r) at different distances (10, 20, 30 and 40cm).
3. Consider two beacons, one at position (0, 0) and one at position (L, 0). Measure ranges r_1 and r_2. You can use the following equations to calculate x and y:

$$x = \left(\frac{L^2+r_1^2-r_2^2}{2L}\right)$$

$$y = \pm\sqrt{\left(r_1^2 - \left(\frac{L^2+r_1^2-r_2^2}{2L}\right)^2\right)}$$

Assume the variance of the ultrasound sensor to be σ^2_r. Calculate an expression for σ^2_x and σ^2_y.

Deliverables

1. Provide a brief write-up showing your data collected in (2) and your solution (equations) to (3).

Data Collected from 10, 20, 30 and 40cm (with the beacon same position and beacon turned wide side showing for working 40cm):

```
104 103 18 -1 -1 -1 -1 -1 -1 -1 -1 17 16 15 15 15 15 15 15 14 14 14 14 14 14
14 14 14 14 14 14 14 14 14 14 15 15 14 15 15 15 15 15 15 15 15 16 17 104
241 104 104 103 102 102 103 103 103 103 102
10cm Variance: 6

166 62 -1 -1 -1 -1 -1 314 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 24 25 25 24 23 24
23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 24 24 24 24 167 168 168
167 168 167 167 167 167 166 167 167 166 166 166
20cm Variance: 6

166 67 76 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 30 30 30 30 30 29
29 29 29 29 28 28 29 28 29 28 28 28 29 28 28 28 28 29 28 28 29 28 29 28 29 28
29 29 29 167 166 166 166 166 166 166
30cm Variance: 10

173 67 -1 -1 642 -1 -1 316 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 44 -1 -1 -1 -1 -1 -1 -1 278 279 278 178 176 177 -1 175 175 175 176
174 176 174 174 174 174 174 174 174 174 174 174 174 174
40cm Variance: None

175 131 43 -1 -1 -1 -1 330 -1 -1 -1 43 43 42 42 42 42 42 42 42 42 43 43 43 43
43 43 43 43 43 43 43 43 43 43 43 43 44 43 43 43 43 43 43 44 44 43 44 44 44 44
44 44 45 174 174 174 174 174 174 173
40cm Variance: 2
```

For step two, we used the equation to compute the variance of a sample set of data:

$$S^2 = \frac{\sum(x_i - \bar{x})^2}{n-1}$$

x_i = Data from data set at instance i

$\bar{x}$ = Sum of all data values in the data set over the total amount of data n, $\frac{\sum x_i}{n}$

n = Total amount of sample values for beacon distance.

We used the variance equation for a sample set and not the population variance equation because our code only collected the first 10 samples data values for the beacon distance out of the entire

varying amount. This allowed for better consistency between our data sets and easier comparison that can allow to determine and approximate the true variance.

We had to recheck the data for 40cm using the wide side of the beacon box, as the beacon was too far away for Sparki to detect. This is most likely due to the distance causing Sparki to miss more ultrasound pings to the beacon, since the surface area of the beacon becomes a smaller target the further it is from the sensor.

For step three, we placed one beacon at the same ~10cm distance from the robot. This will be our coordinate (0, 0). We then placed another one roughly 12cm away from it, labeling the coordinate (12, 0) for our calculations. That second beacon was right to Sparki and positive in terms of the x position.

Here is that data:

```
104 103 18 17 16 15 15 15 15 15 15 14 14 14 14 14 14 14 15 15 14 15 15 15 15
15 15 16 17 104 241 104 104 103 102 17 16 15 15 15 14 14 14 14 13 13 13 12 13
14 14 15 15 16 17 17 103 103 103 102
```

We then calculated the range values r_1^2 and r_2^2:

$r_1^2 = 17 - 14 = 3$

$r_2^2 = 17 - 12 = 5$

Then, using the equations to solve for x and y variances:

$x = \frac{12^2+3^2-5^2}{2*12} = \frac{128}{24}$ = **5.33333333…**

$y = \pm\sqrt{3^2 - \left(\frac{128}{24}\right)^2}$ = **±2.96273147…**

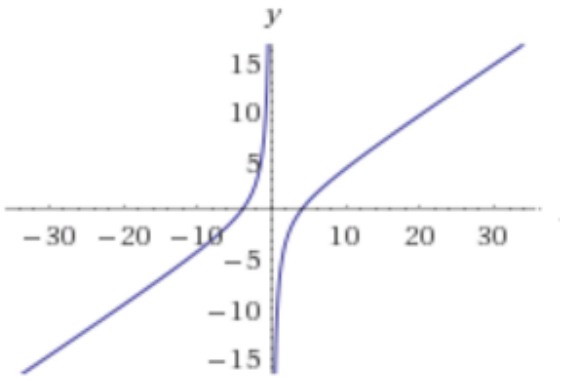
2. What happens to the variance of x and y when L increases?

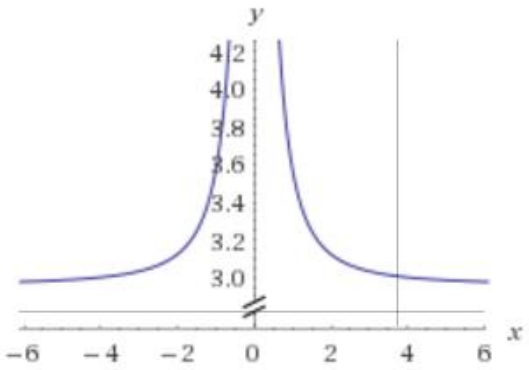
If the variable L were to increase, it would cause an exponential increase in the x variance with a diminishing return into a linear trend the larger L gets, and a decrease in the y variance until y converges into the value of r_1^2.

Below are graphs depicting the change in both the x and y variance with a varying L, given that r_1^2 and r_2^2 are unchanging. In these graphs, both range values are that of the beacons in the data set.

The y axis represents the corresponding value's variance and the x axis represents L.

x y





3. When would you rely on this position estimate over your odometry estimate? When would you rather trust your odometry?

Typically this variant of position estimation can yield an overall more accurate approximation for location as compared to internally calculated odometry. This is because there are physical beacon indicators within the world that are being used to help track and calculate the robot's coordinate information. By placing these static objects within the traversal plane, the robot is able to depend on these markers and can easily maintain orientation or adjust for proper movement. This is less prone to accumulating error and can typically yield more stable position tracking. However, this method creates a new dependency to the world, which cannot be readily applied as Odometry.

Odometry, on the other hand, is handled without any physical markers in this world. Everything done via odometry is done with respect to the robot and its self-maintained data. This means that the robot is more independent from the traversing terrain and doesn't require as many external variables to help orientate itself. This method is more prone to accumulating error, but requires less constants for a similar functionality. Generally, I would trust odometry more in the case of applying a functional robot device within a dynamically changing terrain, as the odometry would yield better results in areas where markers can't remain constant. However, I would refer to the other version of position estimation in more easily maintainable application fields, since these areas allow for more control and can allow for more opportunities of improved accuracy.

4. Describe how you could estimate your bearing θ from your position estimate.

For any similar case of Sparki's ultrasound sensor paired with a beacon, one way estimating θ can be done is through determining the timing of beacon detection. If we were to preserve and record all sensor pings within an iteration of scanning, the robot would be able to determine the distance via the beacon's values while determining its angular orientation through the position of those values within the given data structure.

For example, each iteration of ultrasound detecting for Sparki provides 61 pings as the servo head rotates from -30 degrees to 30 degrees. If we recorded all 61 pings into an array, and the head portion of the array pointed towards the beacon distance values, than that would mean Sparki

detected a beacon at -30 degrees from its current servo orientation. Since the beacons are in a constant position, Sparki can readjust itself to a specific degree and start the next iteration. The array would then be repopulated and the beacon distance values would be in a different position within the array. It would become a matter of determining the first position within the array for that of the beacon distance and then calculating the change in θ given the shift in the array value's positions.