Homework 5

MacMillan, Kyle

November 26, 2018

Contents

List of Figures

1 Chapter 11

1.1 Problem 3

To complete this problem I created a class to generate a random map of any size you want up to 99. It generates a random number of obstacles of random size. Play around with it, it's fun. You can change the random numbers or just run the file multiple times. Code for this problem can be found here.

Demonstration is shown in Figure 1. This is an application of the WaveFront BFS algorithm. From there it goes on to find the shortest path as seen in Figure 2. The seed for that particular example is: `7635686187880284248`

Figure 1: WaveFront Distance Evaluation

Figure 2: WaveFront Shortest Path

I discussed the algorithm with Dr. McGough and he was in agreement that it didn't matter if you begin the wave at the start or goal. I reversed the point of origin for the WaveFront because it makes more sense in my brain. The result is the same so long as you start the WaveFront from one end and walk from the other end back to the WaveFront origin. It's more robust if done this way, that way if the goal is moving you don't have to redo the BFS, you just redo the navigation step.

2 Chapter 16

2.1 Problem 1

Given mesaurements of: $z_1 = 284$, $z_2 = 257$, and $z_3 = 295$ and standard deviations of $\sigma_1 = 10$, $\sigma_2 = 20$, and $\sigma_3 = 15$ we can setup variance as the square of the standard deviations:

1. 100
2. 400
3. 225

Given the variance we can apply the formulas from section 16.3.3 in the book to obtain an estimated state $\hat{x} = 282.90163934426226$. Code for this problem can be found here and in the code snippet 2.1.

```
# Given values
measured = np.array([284, 257, 295])
std = np.array([10, 20, 15])

# Calculate variance
var = std * std

# Sensor fusion to obtain x_hat
top = 0.0
bot = 0.0
for i in range(3):
    top += (measured[i] / var[i])
    bot += 1 / var[i]

# Estimated distance
x_hat = top / bot
```

2.2 Problem 2

Given 40 measurements at 2 meters we can calculate the mean:

A 2.23521735

B 1.86443904

C 2.33806001

That is how far off each senor averages from 2 meters. Each new sensor reading then has that mean subtracted from it to yield:

A 2.22255225

B 2.03233526

C 1.79719609

We can then apply the formula provided in the sensor fusion portion of the book for an expected distance, $\hat{x} = 2.057449909256987$ meters. The code for this can be found here and in the code snippet 2.2.

```
# Calculate mean and standard deviation
mean = np.mean(dist_sens, dtype=np.float64, axis=0)
std = np.std(dist_sens, dtype=np.float64, axis=0)

# Reshape because it doesn't need to be 2D
np.reshape(mean, (1, 3))
np.reshape(std, (1, 3))

# Input for this step
new_sens = np.array([2.4577696, 1.8967743, 2.1352561]) + (2.0 - mean)

# Calculate variance
var = std * std

# Sensor fusion to obtain x_hat
top = 0.0
bot = 0.0
for i in range(3):
    top += (new_sens[i] / var[i])
    bot += 1 / var[i]

# Estimated distance
x_hat = top / bot
```

3 Chapter 17

3.1 Problem 2

asdf

3.2 Problem 3

asdf

3.3 Problem 4

3.3.1 Problem 4a

asdf

3.3.2 Problem 4b

asdf

3.3.3 Problem 4c

asdf

3.3.4 Problem 4d

asdf

3.3.5 Problem 4e

asdf

4 Chapter 18

4.1 Problem 1

4.1.1 Problem 1.1

asdf

4.1.2 Problem 1.2

asdf

4.1.3 Problem 1.3

asdf