Spencer Drach

HW #4

Intro to Robotics

Dr. McGough

1. Assume that you are working in a large event center which has beacons located around the facility. Estimate the location of a robot,(a,b,c), if the(x,y,z)location of the beacon and the distance from the beacon to the robot,d, are given in the table below.

x	y	z	d
884	554	713	222
120	703	771	843
938	871	583	436
967	653	46	529
593	186	989	610

Using the Code in Appendix 1 I came to the following result:

(x,y,z) = (883.1165, 442.99567, 521.41609)

2. If you are using a laser diode to build a distance sensor, you need some method to determine the travel time. Instead of using pulses and a clock, try using phase shifts. What is the wavelength of the modulated frequency of 10MHz?

The wavelength of a 10MHz frequency signal is c/10MHz = 29.979 meters

If you measure a 10 degree phase shift, this value corresponds to what distances?

Find the percentage of the full wavelength changed: 10/360 = 0.02777

multiple the percent by the full wavelength: 29.979 * 0.027777 = 0.83275 meters

What if the phase shift measurement has noise: zero mean with standard deviation 0.1?

0.1/360 = 0.000278% of a full wavelength

29.979 * 0.000278 = 0.008328 meters for 1 sd of error

How does one get a good estimate of position if the ranges to be measured are from 20 meters to 250 meters?

Layering multiple frequencies on the same signal is a good way of getting fine detail from the high frequencies, but maintain the range that a low frequency would give you.

9.1 Assume you have a laser triangulation system as shown in Fig. 9.2 given by (9.1) and that f=8mm, b=30cm. What are the ranges for α and u if we need to measure target distances in a region (in cm) $20<z<100$ and $10<x<30$?

Using the code in appendix 2 I found the following min's and max's for u and α

$0.785 < \alpha < 1.56$ rads and

$0.08 < u < 1.18$ cm

it should be noted that as $x \rightarrow 30$ you get inverse tan of (z/0) so it can't actually reach that edge of the region

9.2 Assume you have two cameras that are calibrated into a stereo pair with a baseline of 10cm, and focal depth of 7mm. If the error is 10% on v1 and v2, v1=2mm and v2=3mm, what is the error on the depth measurement z?Your answer should be a percentage relative to the error free number. Hint: If v1=2 then a 10% error ranges from 1.8 to 2.2. [Although not required, another way to approach this problem is the total differential from calculus.]

Using the code in appendix 3 I found the minimum value to be 127.2727 mm. I found the maximum value to be 155.5555 mm and the no-error value to be 140 mm. Giving me a % error of 9.09% for the min value and 11.11 for the max value.

Appendix 1: Code for chapter 8 problem 1

```
def Problem1():

    import numpy as np
    import scipy as sp
    import matplotlib as mpl
    import matplotlib.pyplot as plt
    import math as math
    # from math import *


    # Location of the beacons and distances
    L = [[884,554,713,222],[120,703,771,843],[938,871,583,436],[967,653,46,529],
[593,186,989,610]]

    def funct(x,y,z):
    F = np.sqrt((x-L[0][0])*(x-L[0][0]) + (y-L[0][1])*(y-L[0][1]) + (z-L[0][2])*(z-L[0][2])) - L[0][3]
    G = np.sqrt((x-L[1][0])*(x-L[1][0]) + (y-L[1][1])*(y-L[1][1]) + (z-L[1][2])*(z-L[1][2])) - L[1][3]
    H = np.sqrt((x-L[2][0])*(x-L[2][0]) + (y-L[2][1])*(y-L[2][1]) + (z-L[2][2])*(z-L[2][2])) - L[2][3]
    I = np.sqrt((x-L[3][0])*(x-L[3][0]) + (y-L[3][1])*(y-L[3][1]) + (z-L[3][2])*(z-L[3][2])) - L[3][3]
    K = np.sqrt((x-L[4][0])*(x-L[4][0]) + (y-L[4][1])*(y-L[4][1]) + (z-L[4][2])*(z-L[4][2])) - L[4][3]

        F=F*F
        G=G*G
        H=H*H
        I=I*I
        K=K*K

        E = F+G+H+I+K
        # print(E)
        return E

    # Numerical gradient approximation
    def grad(x,y,z):
        delta = 0.0001
        E = funct(x,y,z)
        E1 = funct(x+delta,y,z)
        E2 = funct(x,y+delta,z)
        E3 = funct(x,y,z+delta)
        dEx = (E1-E)/delta
        dEy = (E2-E)/delta
        dEz = (E3-E)/delta
        return dEx, dEy, dEz


    # The size of the vector
    def norm(r,s,q):
        return np.sqrt(r*r+s*s+q*q)

    # The step in the direction (u,v)
    def step(x,y,z, u,v,w,t):
        a = x - t*u
        b = y - t*v
        c = z - t*w
```

```
        return a, b, c


    # Globals
    x = 5000
    y = 5000
    z = 5000
    t = 20.0
    tsmall = 0.00001

    # The descent algorithm
    while (t > tsmall):
        dx, dy, dz = grad(x,y,z)
        size = norm(dx,dy,dz)
        u = dx/size
        v = dy/size
        w = dz/size
        a,b,c = step(x,y,z,u,v,w,t)
        while (funct(a,b,c) > funct(x,y,z)):
            t = 0.5*t
            a,b,c = step(x,y,z,u,v,w,t)
        x,y,z = a,b,c

    print(x, y, z)

Problem1()
```

Appendix 2: Chapter 9 problem 1

```
def Problem12():

    import numpy as np
    import scipy as sp
    import matplotlib as mpl
    import matplotlib.pyplot as plt
    import math as math

    z = [20,20,100,100]
    x = [10,29,10,29]

    f = 0.8
    b = 30

    for i in range(len(z)):
        u = f/(z[i]/x[i])
        alpha = np.arctan(z[i]/(b-x[i]))
        print(u,alpha)



Problem12()
```

Appendix 3: Chapter 9 problem 2

```
def Problem22():

    import numpy as np
    import scipy as sp
    import matplotlib as mpl
    import matplotlib.pyplot as plt
    import math as math

    f = 7
    b = 100

    v1 = [2.0,2.2,2.2,1.8,1.8]
    v2 = [3.0,3.3,2.7,3.3,2.7]
    z = []

    for i in range(len(v1)):
        z.append((f*b)/(v1[i]+v2[i]))

    z_min = min(z)
    z_max = max(z)
    error_min = 100 * (z_min-z[0])/z[0]
    error_max = 100 * (z_max-z[0])/z[0]
    print("actual value: ", z[0])
    print("min value: ", z_min, "max value: ", z_max)
    print("error min: ", error_min, "error max: ", error_max)

Problem22()
```