海中ロボット学

濱松祐矢

from math import \*

import random

import numpy as np

import matplotlib.pyplot as plt

landmarks = [[0.0, 10.0]]

world\_size = 200.0

t = 0 # sec

class robot:

def \_\_init\_\_(self):

self.x = np.random.normal(0, 2.0)

self.y = np.random.normal(0, 1.0)

self.orientation = pi /2 + np.random.normal(0, 0.026)

self.x\_noise = 0.0;

self.y\_noise = 0.0;

self.yaw\_noise = 0.0;

self.sense\_noise = 0.0;

def set(self, new\_x, new\_y, new\_orientation):

self.x = float(new\_x)

self.y = float(new\_y)

self.orientation = float(new\_orientation)

def set\_noise(self, new\_x\_noise, new\_y\_noise, new\_yaw\_noise, new\_sense\_noise):

self.x\_noise = float(new\_x\_noise);

self.y\_noise = float(new\_y\_noise);

self.yaw\_noise = float(new\_yaw\_noise);

self.sense\_noise = float(new\_sense\_noise);

def sense(self):

Z = []

for i in range(len(landmarks)):

dist = np.sqrt((self.x - landmarks[i][0]) \*\* 2 + (self.y - landmarks[i][1]) \*\* 2)

dist \*= np.random.normal(0.0, self.sense\_noise)

Z.append(dist)

return Z

def move(self, turn, forward):

orientation = self.orientation + float(turn) + np.random.normal(0.0, self.yaw\_noise)

orientation %= 2 \* pi

dist = float(forward)

x = self.x + (np.cos(orientation) \* dist) + np.random.normal(0.0, self.x\_noise)

y = self.y + (np.sin(orientation) \* dist) + np.random.normal(0.0, self.y\_noise)

# set particle

res = robot()

res.set(x, y, orientation)

res.set\_noise(self.x\_noise, self.y\_noise, self.yaw\_noise, self.sense\_noise)

return res

def Gaussian(self, mu, sigma, x):

# calculates the probability of x for 1-dim Gaussian with mean mu and var. sigma

return exp(- ((mu - x) \*\* 2) / (sigma \*\* 2) / 2.0) / sqrt(2.0 \* pi \* (sigma \*\* 2))

def measurement\_prob(self, measurement):

# calculates how likely a measurement should be

prob = 1.0 ;

for i in range(len(landmarks)):

dist = np.sqrt((self.x - landmarks[i][0]) \*\* 2 + (self.y - landmarks[i][1]) \*\* 2) + np.random.normal(0.0, self.sense\_noise)

# prob \*= self.Gaussian(dist, self.sense\_noise, measurement[i])

return dist

def \_\_repr\_\_(self):

return '[x=%.6s y=%.6s orient=%.6s]' % (str(self.x), str(self.y), str(self.orientation))

N = 1000

p = []

myrobot = robot()

myrobot.set(0.0, 0.0, pi / 2)

#for i in range(N):

# p[i].set(0.0, 0.0, pi / 2)

for i in range(N):

x = robot()

x.set\_noise(0.1, 0.1, 0.026, 0.5) # x,y,rad,sense

p.append(x)

for i in range(100):

myrobot = myrobot.move(0.0, 1.0)

# myrobot.set\_noise(0.2, 0.2, 0.026, 0.5)

p2 = []

for i in range(N):

p2.append(p[i].move(0.0, 1.0))

p = p2

if t % 10 == 0 :

for i in range(N):

plt.plot(p[i].x, p[i].y, '.')

t =t + 1

print(t)

plt.ylim(-20, 130)

plt.xlim(-50, 50)

plt.show()

----Run----

runfile('C:/Users/hama6767/Documents/underwater\_robotics\_5\_particle\_filter/pf1.py', wdir='C:/Users/hama6767/Documents/underwater\_robotics\_5\_particle\_filter')

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

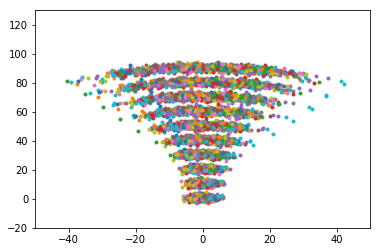
96

97

98

99

100



from math import \*

import random

import numpy as np

import matplotlib.pyplot as plt

landmarks = [[0.0, 10.0]]

world\_size = 200.0

t = 0 # sec

class robot:

def \_\_init\_\_(self):

self.x = np.random.normal(0, 1.0)

self.y = np.random.normal(0, 1.0)

self.orientation = pi /2 + np.random.normal(0, 0.026)

self.x\_noise = 0.2;

self.y\_noise = 0.4;

self.yaw\_noise = 0.026;

self.sense\_noise = 0.0;

def set(self, new\_x, new\_y, new\_orientation):

self.x = float(new\_x)

self.y = float(new\_y)

self.orientation = float(new\_orientation)

def set\_noise(self, new\_x\_noise, new\_y\_noise, new\_yaw\_noise, new\_sense\_noise):

# makes it possible to change the noise parameters

# this is often useful in particle filters

self.x\_noise = float(new\_x\_noise);

self.y\_noise = float(new\_y\_noise);

self.yaw\_noise = float(new\_yaw\_noise);

self.sense\_noise = float(new\_sense\_noise);

def sense(self):

Z = []

for i in range(len(landmarks)):

dist = np.sqrt((self.x - landmarks[i][0]) \*\* 2 + (self.y - landmarks[i][1]) \*\* 2)

dist \*= np.random.normal(0.0, self.sense\_noise)

Z.append(dist)

return Z

def move(self, turn, forward):

# turn, and add randomness to the turning command

orientation = self.orientation + float(turn) + np.random.normal(0.0, self.yaw\_noise)

orientation %= 2 \* pi

# move, and add randomness to the motion command

dist = float(forward)

x = self.x + (np.cos(orientation) \* dist) + np.random.normal(0.0, self.x\_noise)

y = self.y + (np.sin(orientation) \* dist) + np.random.normal(0.0, self.y\_noise)

# set particle

res = robot()

res.set(x, y, orientation)

res.set\_noise(self.x\_noise, self.y\_noise, self.yaw\_noise, self.sense\_noise)

return res

def Gaussian(self, mu, sigma, x):

# calculates the probability of x for 1-dim Gaussian with mean mu and var. sigma

return exp(- ((mu - x) \*\* 2) / (sigma \*\* 2) / 2.0) / sqrt(2.0 \* pi \* (sigma \*\* 2))

def measurement\_prob(self, measurement):

# calculates how likely a measurement should be

prob = 1.0 ;

for i in range(len(landmarks)):

dist = np.sqrt((self.x - landmarks[i][0]) \*\* 2 + (self.y - landmarks[i][1]) \*\* 2) + np.random.normal(0.0, self.sense\_noise)

# prob \*= self.Gaussian(dist, self.sense\_noise, measurement[i])

return dist

def \_\_repr\_\_(self):

return '[x=%.6s y=%.6s orient=%.6s]' % (str(self.x), str(self.y), str(self.orientation))

myrobot = robot()

myrobot.set(0.0, 0.0, pi/2)

N = 700

p = []

for i in range(N):

x = robot()

x.set\_noise(1.0, 1.0, 0.026, 0.5) # x,y,rad,sense

p.append(x)

for i in range(100):

myrobot = myrobot.move(0.0, 0.0)

myrobot.set\_noise(1.0, 1.0, 0.026, 0.5)

p2 = []

for i in range(N):

p2.append(p[i].move(0.0, 0.0))

p = p2

if t % 10 ==0 :

Z = myrobot.sense()

w = []

wp = 0

for i in range(N):

w.append(p[i].measurement\_prob(Z))

p3 = []

p3.append(p[1])

for i in range(N-1):

if abs(10.0 - w[i]) > abs(10.0 - w[i+1]) :

p3.append(p[i+1])

else:

if wp < 7:

p3.append(p[i])

p[i+1].x = p[i].x #+ np.random.normal(0.0, 0.05)

p[i+1].y = p[i].y #+ np.random.normal(0.0, 0.05)

w[i+1] = w[i]

wp += 1

else:

p3.append(p[i])

wp = 0

p = p3

random.shuffle(p)

if t % 10 == 0 :

for i in range(N):

plt.plot(p[i].x, p[i].y, '.')

t =t + 1

print(t)

plt.xlim(-20, 20)

plt.ylim(-20, 20)

plt.show()

----Run-----

runfile('C:/Users/hama6767/Documents/underwater\_robotics\_5\_particle\_filter/pf.py', wdir='C:/Users/hama6767/Documents/underwater\_robotics\_5\_particle\_filter')

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

