

計算型智慧

作業二

模糊系統

學號：109401553


系級：資管三 B

學生：楊雲杰

日期：2024/05/21

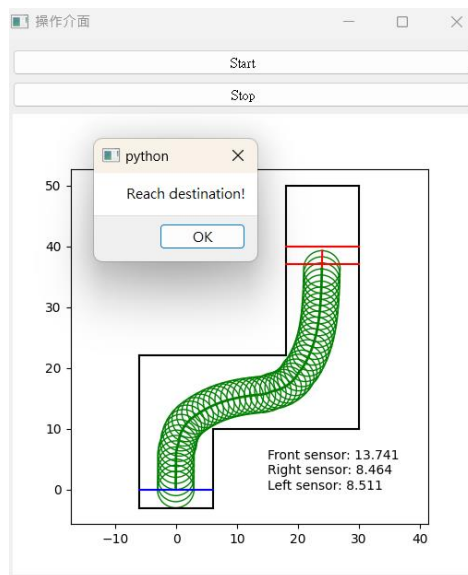
一、程式介面說明

我的執行檔路徑為 exe_file/simple_playground/simple_playground

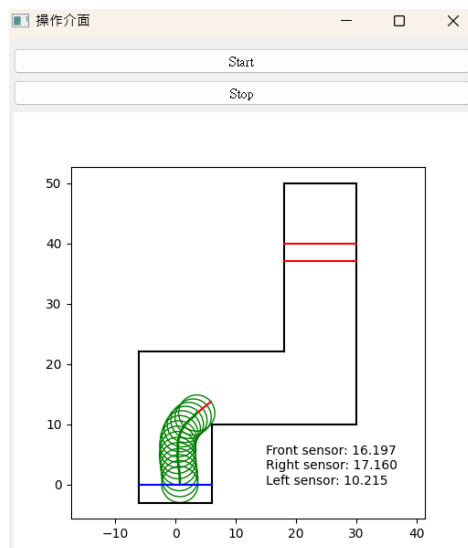
 _internal	2024/5/19 下午 09:51	檔案資料夾	
 simple_playground	2024/5/19 下午 09:51	應用程式	5,111 KB

圖一、執行檔路徑

起點線為藍色線，終點為紅色長方形，車子為綠色空心圓圈(為了方便看出軌跡)。介面上方有按鈕 Start，點擊即可讓車子進行一次模擬。Stop 則是在模擬進行中可以點擊暫停觀察狀況。介面右下角另有三行文字，分別為正前方、右 45 度角、左 45 度角的感測器所測量出的距離。



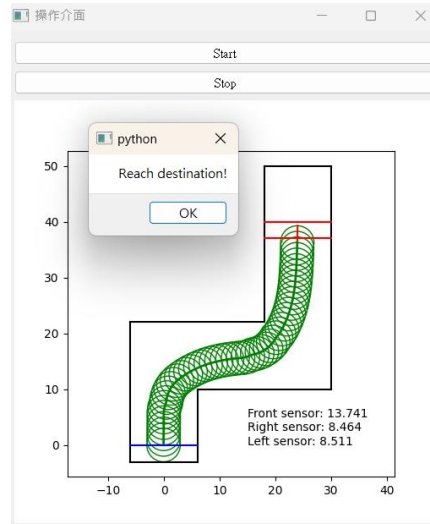
圖二、成功畫面



圖三、失敗畫面

二、實驗結果

依據題目要求，我使用左側、前方、右側三個感測器實作模糊系統，成功讓自走車一次抵達終點。另外，經過 1000 次測試，我的自走車有約 97% 的機率可以一次到達終點(詳見圖五)。



圖四、一次走到終點

```
PS C:\Users\User\Desktop\大學課程內容\計算型智慧 Computational Intelligence (蘇木春)\109401553_楊雲杰_計算型智慧_作業二> & C:/Users/User/AppData/Local/Programs/Python/Python39/python.exe "c:/Users/User/Desktop/大學課程內容/計算型智慧 Computational Intelligence (蘇木春)/109401553_楊雲杰_計算型智慧_作業二/simple_playground.py"
Success rate: 96.60%
```

圖五、1000 次測試結果

三、歸屬函數說明

歸屬函數說明的部分我會先說明我是如何實作，再將歸屬函數圖畫出。(相關程式碼可以對照 fuzzy.py 以及 membership_function.py，fuzzy.py 主要是實作模糊系統，membership_function.py 主要是畫出歸屬函數的樣子)。以下是模糊系統之 If-Then 規則(圖八)：

Rule 1: If front is Near and more space is on the left, then turn Left.

Rule 2: If front is Near and space is balanced, then go Straight.

Rule 3: If front is Near and more space is on the right, then turn Right.

Rule 4: If front is Medium and more space is on the left, then turn Left.

Rule 5: If front is Medium and space is balanced, then go Straight.

Rule 6: If front is Medium and more space is on the right, then turn Right.

Rule 7: If front is Far and more space is on the left, then turn Left.

Rule 8: If front is Far and space is balanced, then go Straight.

Rule 9: If front is Far and more space is on the right, then turn Right.

```

10     def front_near(x):
11         if x <= 4:
12             return 1
13         elif 4 < x <= 5:
14             return (5 - x) / 1
15         else:
16             return 0
17
18     def front_medium(x):
19         if 6 < x < 9:
20             return 1
21         elif 4 <= x <= 6:
22             return (x - 4) / 2
23         elif 9 <= x <= 11:
24             return (11 - x) / 2
25         else:
26             return 0
27
28     def front_far(x):
29         if x > 12:
30             return 1
31         elif 10 <= x < 12:
32             return (x - 10) / 2
33         else:
34             return 0

```

圖六、正前方感測器歸屬函數

```

36     def right_left_diff_left(y):
37         if y <= -4:
38             return 1
39         elif -4 < y <= 0:
40             return (-y) / 4
41         else:
42             return 0
43
44
45     def right_left_diff_straight(y):
46         if -1 <= y <= 1:
47             return 1 - abs(y)
48         else:
49             return 0
50
51     def right_left_diff_right(y):
52         if y >= 4:
53             return 1
54         elif 0 <= y < 4:
55             return y / 4
56         else:
57             return 0

```

圖七、右側減左側距離的歸屬函數

```

72     # 模糊化輸入
73     front_near_val = front_near(front_distance)
74     front_medium_val = front_medium(front_distance)
75     front_far_val = front_far(front_distance)
76
77     diff = right_distance - left_distance
78     diff_left_val = right_left_diff_left(diff)
79     diff_straight_val = right_left_diff_straight(diff)
80     diff_right_val = right_left_diff_right(diff)
81
82     # 模糊推理規則
83     rule1 = min(front_near_val, diff_left_val)
84     rule2 = min(front_near_val, diff_straight_val)
85     rule3 = min(front_near_val, diff_right_val)
86     rule4 = min(front_medium_val, diff_left_val)
87     rule5 = min(front_medium_val, diff_straight_val)
88     rule6 = min(front_medium_val, diff_right_val)
89     rule7 = min(front_far_val, diff_left_val)
90     rule8 = min(front_far_val, diff_straight_val)
91     rule9 = min(front_far_val, diff_right_val)

```

圖八、模糊規則及輸入

```

62     # 定義輸出的模糊集合
63     def turn_left(z):
64         return np.where(z <= -self.max_delta, 1, np.where(z < 0, -z / self.max_delta, 0))
65
66     def turn_straight(z):
67         return np.where(abs(z) <= 10, (10 - abs(z)) / 10, 0)
68
69     def turn_right(z):
70         return np.where(z >= self.max_delta, 1, np.where(z > 0, z / self.max_delta, 0))
71

```

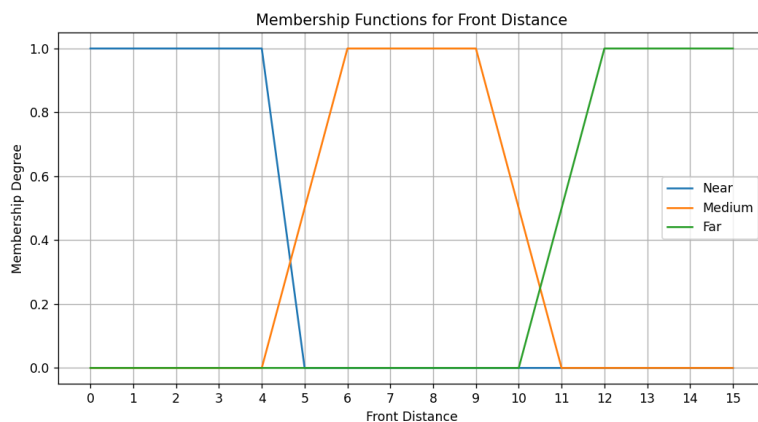
圖九、輸出

1. 正前方感測器接收距離之歸屬函數(可參考圖六)

藍色：Near(<5)

橘色：Medium(4~11)

綠色：Far(>10)



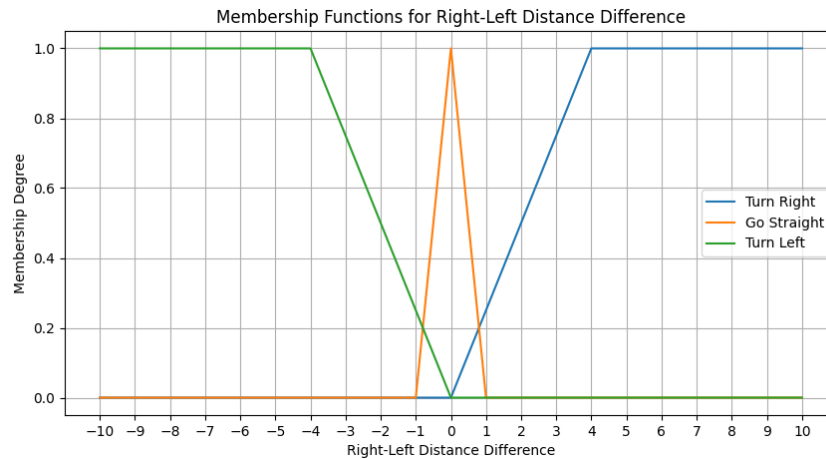
圖十、正前方感測器

2. 右側感測器距離值減左側感測器距離值之歸屬函數(可參考圖七)

藍色：右側有空間(向右轉)

橘色：直走

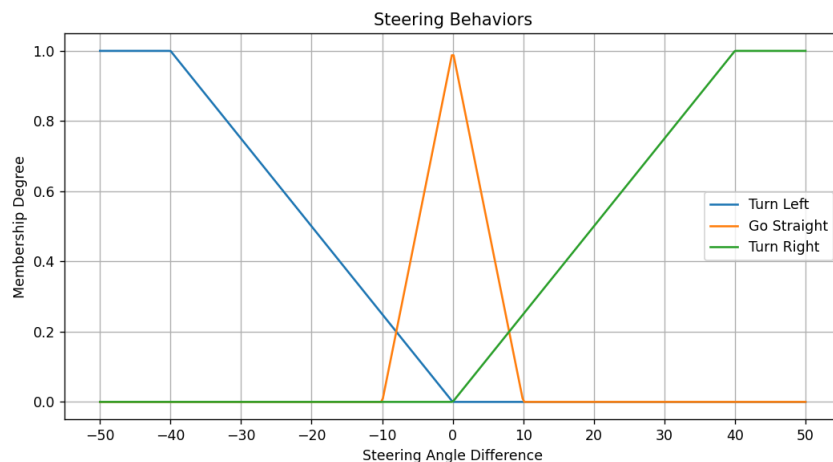
綠色：左側有空間(向左轉)



圖十一、右側減左側感測器距離

3. 輸出：轉向的角度(可參考圖九)

顯示出車子接下來的轉向為哪個方向，其中我採用的去模糊化方法是重心法(可參考 fuzzy.py)。



圖十二、轉向歸屬函數

四、分析

在寫這次作業的過程中，模糊規則的設計以及歸屬函數的撰寫以及去模糊化的規則是我認為非常有挑戰性的部分。

我將距離分成遠、中、近三個部分，左右距離設為左、中、右(左代表左邊有空間，右代表右邊有空間)三個部分。因此，我總共設計了 $9(3*3)$ 個 If-Then 邏輯。而歸屬函數的設計是一個非常繁瑣的過程，我撰寫許多 If, else 的邏輯，並且為了做出較理想的歸屬函數，我花了許多時間進行測試。

至於去模糊化的方式我採用重心法(離散)，因為我發現這個方法得出來的結果較為平穩。其他方法如最大平均法、修正型最大平均法容易出現左右較大擺盪。雖然走到終點的機率也蠻高的，但是經過測試我認為重心法是最理想的方法。

總而言之，這份作業花了我非常多的時間及心血，實現自走車可以走到終點的任務。也希望未來我能利用本次作業所學，將模糊系統應用在各種不同的地方。