به نام خدا

دانشگاه تهران

پردیس دانشکده‌های فنی

دانشکده مهندسی برق و کامپیوتر

مبانی مکاترونیک

استاد: دکتر طالع ماسوله

مینی پروژه 6

فربد سیاه‌کلی

۸۱۰۱۹۸۵۱۰

مرداد ۱۴۰۱

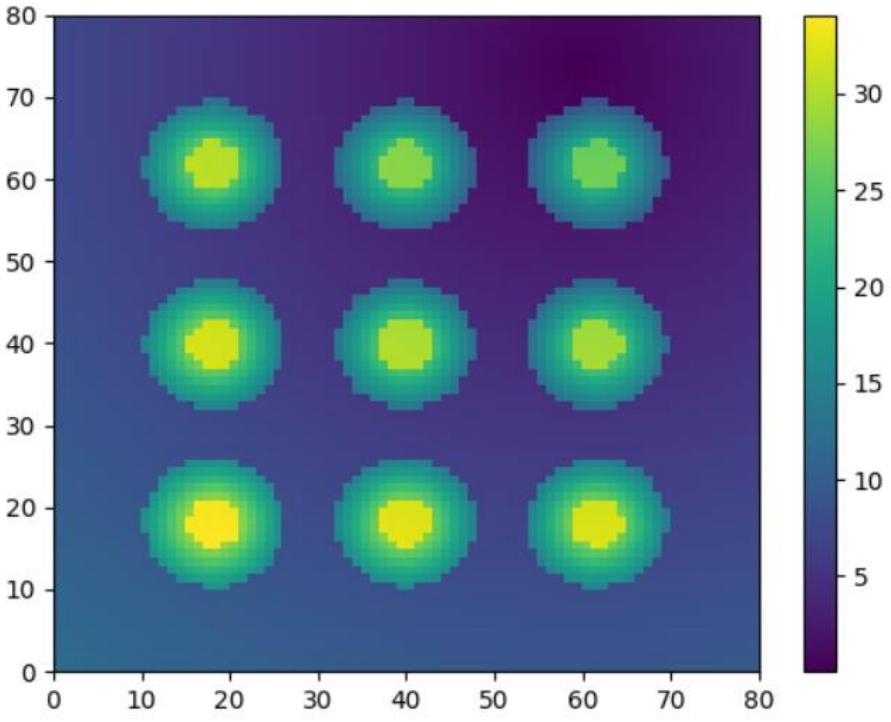
فهرست گزارش سوالات (لطفاً پس از تکمیل گزارش، این فهرست را به‌روز کنید.)

نقشه پتانسیل محیط

محیط در بازه منفی 2 تا 2 به صورت ماتریس های 80 در 80 گسسته سازی شده است تا پتانسیل نقاط محاسبه شوند. پس از محاسبه تاثیر پتانسیل موانع و نقطه هدف بر روی محیط، اقدام به یافتن مسیر بهینه می‌کنیم. (در کد پتانسیل بینهایت برابر با مقدار 25 قرار گرفته تا در Pcolor پتانسیل بقیه نقاط نیز بهتر مشاهده شوند.)

```
alpha = 5
beta = 100
s = 0.4

obstacles = [[-1.1, -1.1], [-1.1, 0], [-1.1, 1.1] , [0, -1.1], [0,0] , [0, 1.1], [1.1, -1.1], [1.1, 0], [1.1, 1.1]]

r = 0.15
l = 0.5

for i, xx in enumerate(x):
    for j, yy in enumerate(y):
        for ob in obstacles:
            goal_potential[i][j] = 0.5*alpha*math.sqrt(pow(xx-goal[0],2) + pow(yy-goal[1],2))
            if (math.sqrt(pow(xx-ob[0],2) + pow(yy-ob[1],2)))<r:
                blocks_potential[i][j] = 25

            elif (math.sqrt(pow(xx-ob[0],2) + pow(yy-ob[1],2)))>s:
                blocks_potential[i][j] += 0

            else:
                blocks_potential[i][j] +=  0.5*beta*(s+r-math.sqrt(pow(xx-ob[0],2) + pow(yy-ob[1],2)))

potential = blocks_potential+goal_potential
```

تصویری از پتانسیل محیط:



مسیریابی به نقطه (1.7, 1)

حال از نقطه آغازین شروع می‌کنیم و پتانسیل نقاط مجاور را بررسی کرده و نهایتا به نقطه‌ای که کمترین پتانسیل را دارد حرکت می‌کنیم. این الگوریتم را تا زمانی اجرا می‌کنیم که به مقصد برسیم.

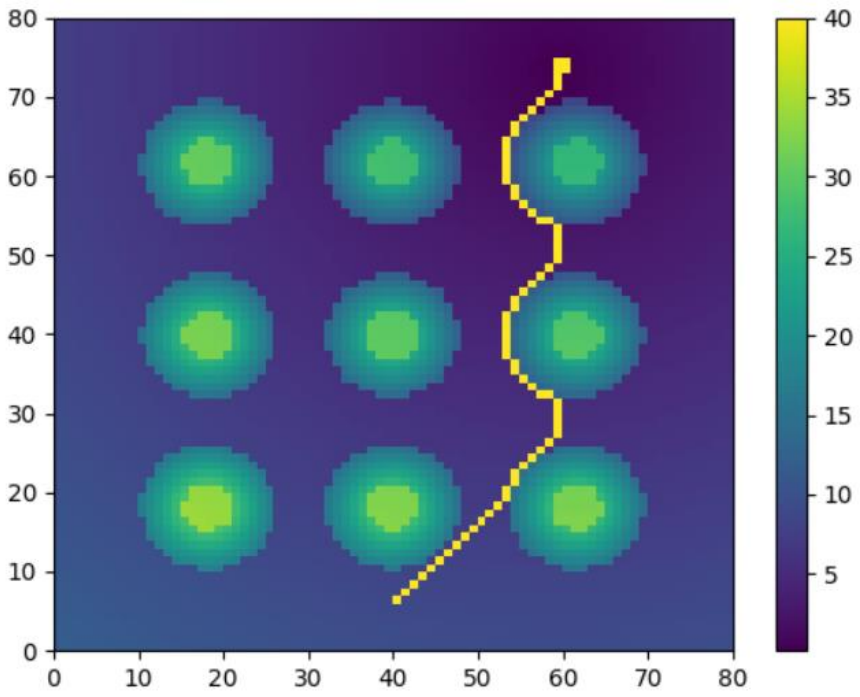
یافتن همسایه‌های یک نقطه:

```
from itertools import product

def neighbours(cell):
    for c in product(*(range(n-1, n+2) for n in cell)):
        if c != cell and all(0 <= n < 80 for n in c):
            yield c
```

پیاده سازی الگوریتم:

```
for near in nears:
    if flag == 0:
        temp = potential[near[0]][near[1]]
        temp_index = near
        flag = 1

    else:
        if potential[near[0]][near[1]] <= temp:
            if near == path[-2] or near == path[-3]:
                continue
            temp = potential[near[0]][near[1]]
            temp_index = near

path.append(temp_index)
pos_now = temp_index
```

نتیجه نهایی:



اعمال مسیر در محیط Gazebo

از آنجیی که هر نقطه، هشت مجاور دارد، باید پس از فهمیدن جهت مدنظر برای حرکت، زاویه ربات را اصلاح کرده و همچنین به میزان مورد نیاز رو به جلو حرکت کنیم.

بعنوان مثال اگر می‌خواهیم به سمت جلو حرکت کنیم زاویه ربات باید صفر درجه باشد و مقدار $\frac{2}{80}$ واحد جلو برویم. برای حرکت اریب نیز باید $\sqrt{2}$ این مقدار را در زاویه مدنظر اعمال کنیم.

کد این بخش به صورت زیر خواهد بود:

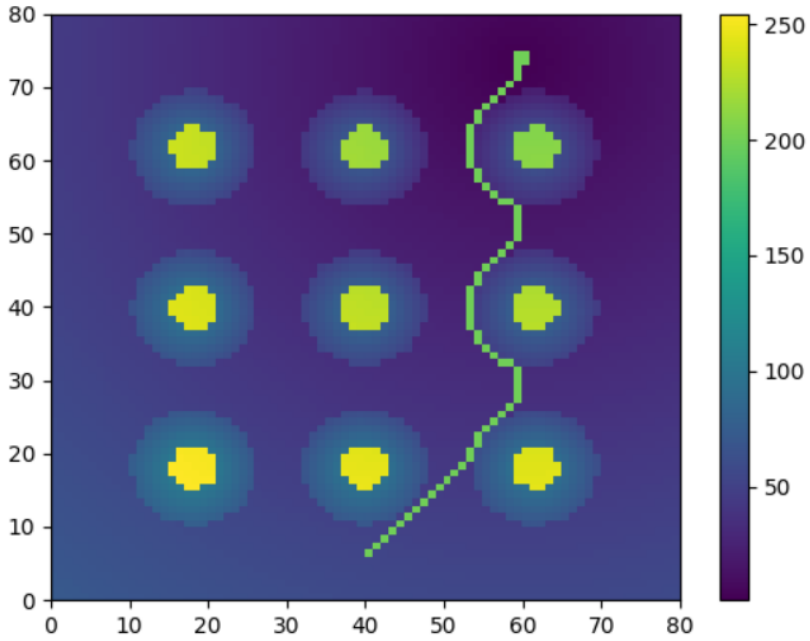
```
print("founded a route")
print("Starting to move...")

for i, p in enumerate(path):
    diff_y = path[i+1][0]-p[0]
    diff_x = path[i+1][1]-p[1]

    if diff_y == 1 and diff_x == 0: #(0,1) up
        if robot.curr_ang_pose == 0:
            robot.move(0.05)
        else:
            robot.rotate(-robot.curr_ang_pose)
            robot.move(0.05)

    elif diff_y == 1 and diff_x == 1: #(1,1) up right
        if robot.curr_ang_pose == -np.pi/4:
            robot.move(0.07071)
        else:
            robot.rotate(-np.pi/4-robot.curr_ang_pose)
            robot.move(0.07071)

    elif diff_y == 1 and diff_x == -1: #(-1,1) up left
        if robot.curr_ang_pose == np.pi/4:
            robot.move(0.07071)
        else:
            robot.rotate(np.pi/4-robot.curr_ang_pose)
            robot.move(0.07071)

    elif diff_y == 0 and diff_x == 1: #(1,0) right
        if robot.curr_ang_pose == -np.pi/2:
            robot.move(0.05)
        else:
            robot.rotate(-np.pi/2-robot.curr_ang_pose)
            robot.move(0.05)

    elif diff_y == 0 and diff_x == -1: #(-1,0) left
        if robot.curr_ang_pose == np.pi/2:
            robot.move(0.05)
        else:
            robot.rotate(np.pi/2-robot.curr_ang_pose)
            robot.move(0.05)

print("Reached at the destination.")
```

که متاسفانه پس از تلاش بسیار(اعم از اعمال دیلی و ...)، ارتباط با گزبو فراهم نشد

تغییر مقادیر α و β

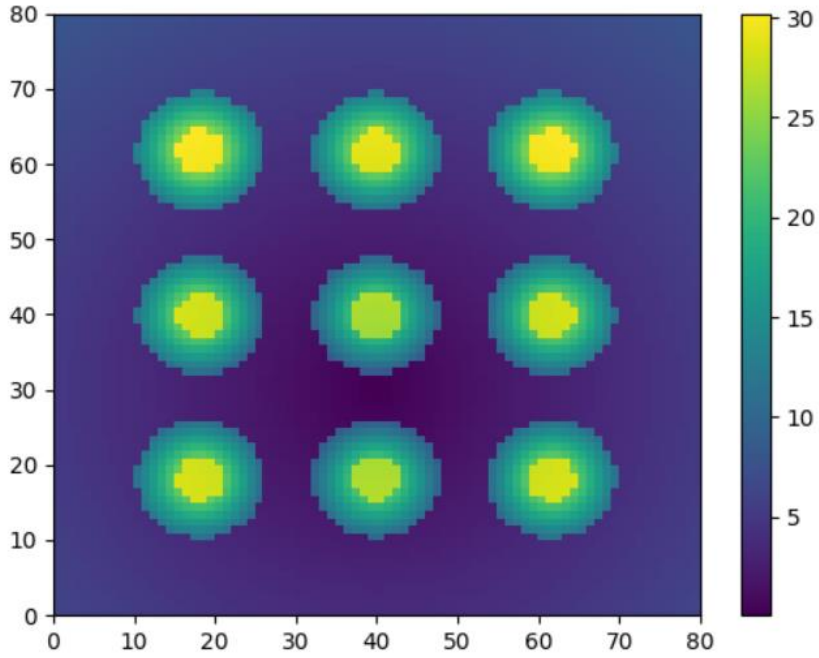
حال مقدار $\alpha = 30$ و $\beta = 250$ را قرار می‌دهیم و الگوریتم را تکرار می‌کنیم.

خروجی به صورت زیر است:



مسیریابی به نقطه (-0.5, 0) به ازای $\alpha = 5$ و $\beta = 100$

پتانسیل فضا:



مسیر یافته شده به صورت زیر خواهد شد: