山东大学

崇新学堂

2022－2023 学年第一学期

实 验 报 告

课程名称：EECS

实验名称：Designlab13

专 业 班 级：崇新 21

学 生 姓 名：张原 刘浩 施政 池弋

实 验 时 间：2022.12.11

Step1 Understand how to generate maps

We tested the setCell function and found that it would produce a 7*7 gray grid with a black body in the middle, as shown in the figure below:

```
def __init__(self, xMin, xMax, yMin, yMax, gridSquareSize):
    self.startState = dynamicGridMap.DynamicGridMap(xMin, xMax, yMin, yMax, gridSquareSize)
def getNextValues(self, state, inp):
    state.setCell((25,25))
    return (state,state)
```

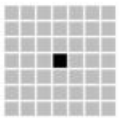

Figure 1 setCell

After testing, we only need to find out the black barrier, and gray barriers are automatically marked. This map is to obtain the position grid at the end of each sonar and set it as barrier.

Step2 Implement the MapMaker class

According to the requirements, first set the starting state, which is a dynamicGridMap.DynamicGridMap class :

```
def __init__(self, xMin, xMax, yMin, yMax, gridSquareSize):
    self.startState = dynamicGridMap.DynamicGridMap(xMin, xMax, yMin, yMax, gridSquareSize)
```

Figure 2 startState

Write the getnextvalues procedure again: it takes the index of the coordinates at the end of the sonar with valid values through the function pointToindices() and marks it as a barrier.

The code is as follows:

```
def getNextValues(self, state, inp):
    sonar=inp.sonars
    pose=inp.odometry
    for i in range(len(sonarDist.sonarPoses)):
        if sonar[i]< sonarDist.sonarMax:#确定声纳探测值为有效的
            black=state.pointToIndices(sonarDist.sonarHit(sonar[i],sonarDist.sonarPoses[i],pose))
            state.setCell(black)
    return (state,state)
```

Figure 3 my code

Step3 Test map maker

Test our map maker inside idle by doing this:

testMapMaker(testData)

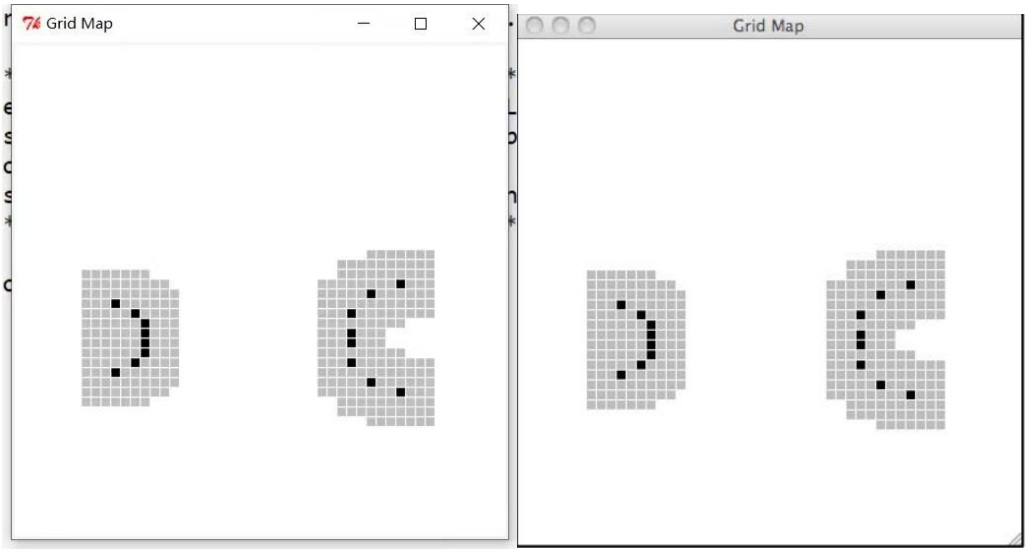

Figure 4 our map and lab map

The code generated map is the same as the handout map, and we're pretty much done.

Step4. test your code in soar

Running the code in soar and testing the car,and results are as follows:

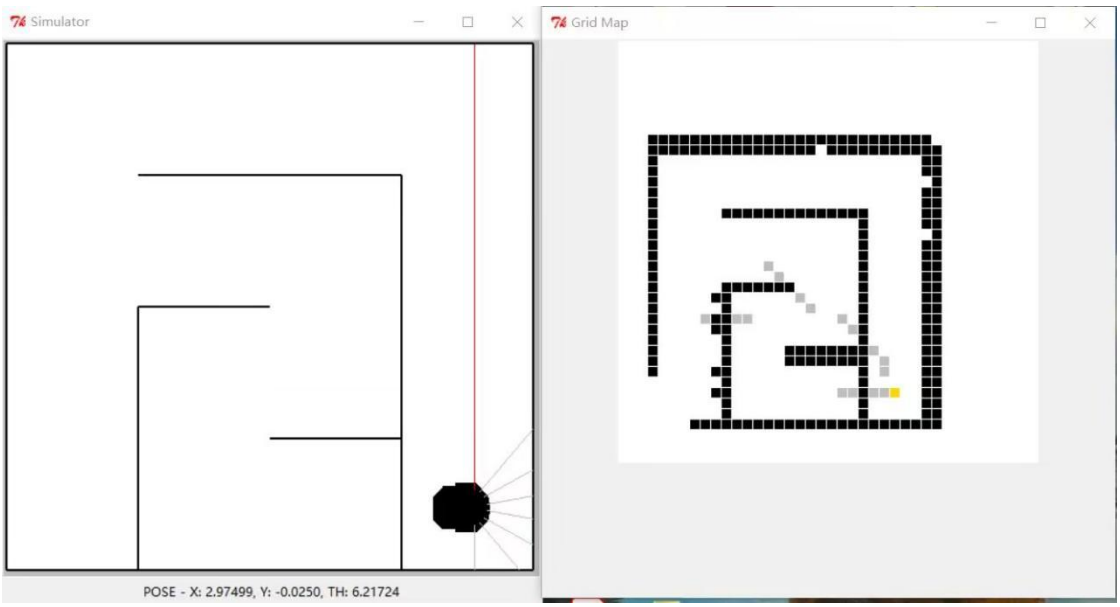

Figure 5 test in soar

Checkoff 1.Show the map and explain it

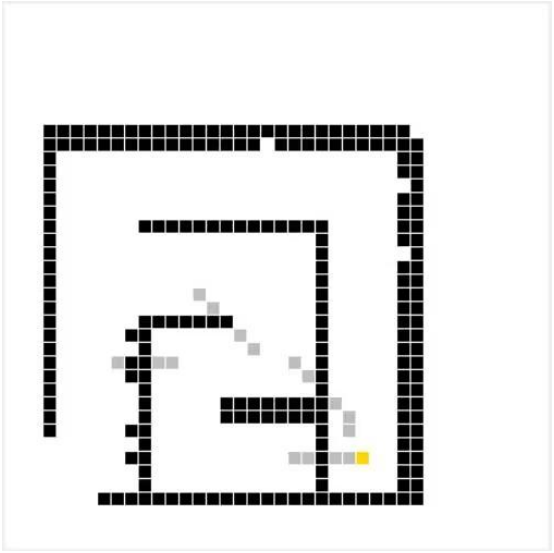

Figure 6 map

Explanation: With each step, the map will be updated with black obstacles, passing the newly generated map to Replanner, which then modifies and selects the nearest path. The following two graphs show that the path is being optimized:

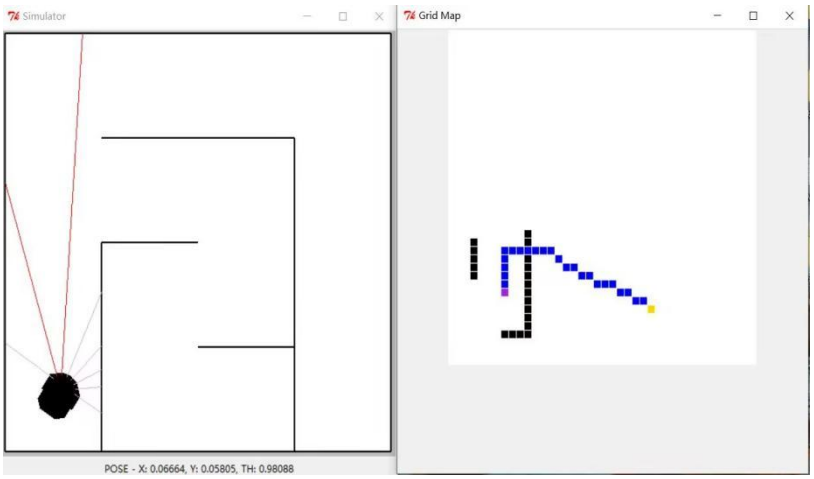

Figure 7 Initial path

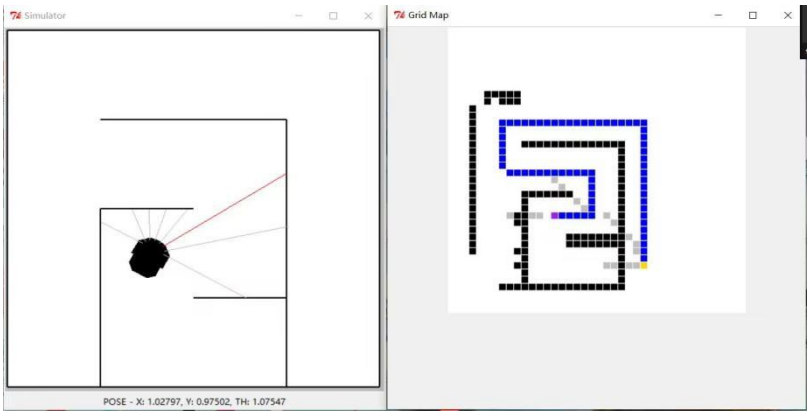

Figure 8 optimized path

Step5 and Check Yourself 2:Set noise and run in soar

Find the line in mapAndReplanBrain.py:

soar.outputs.simulator.SONAR_VARIANCE = lambda mean: noNoise

and change it to this:

soar.outputs.simulator.SONAR_VARIANCE = lambda mean: mediumNoise

This change changed the noise-free environment to a moderately noisy one, and running the car in this environment, we found that the program got stuck and achieved its goal.

How does the noise in the sensor readings affect its performance?

We believe that the noise causes the distance detected by sonar to deviate from the actual distance, and the generated black barrier is inaccurate when the map is updated, which seals the path, as shown in the following figure：

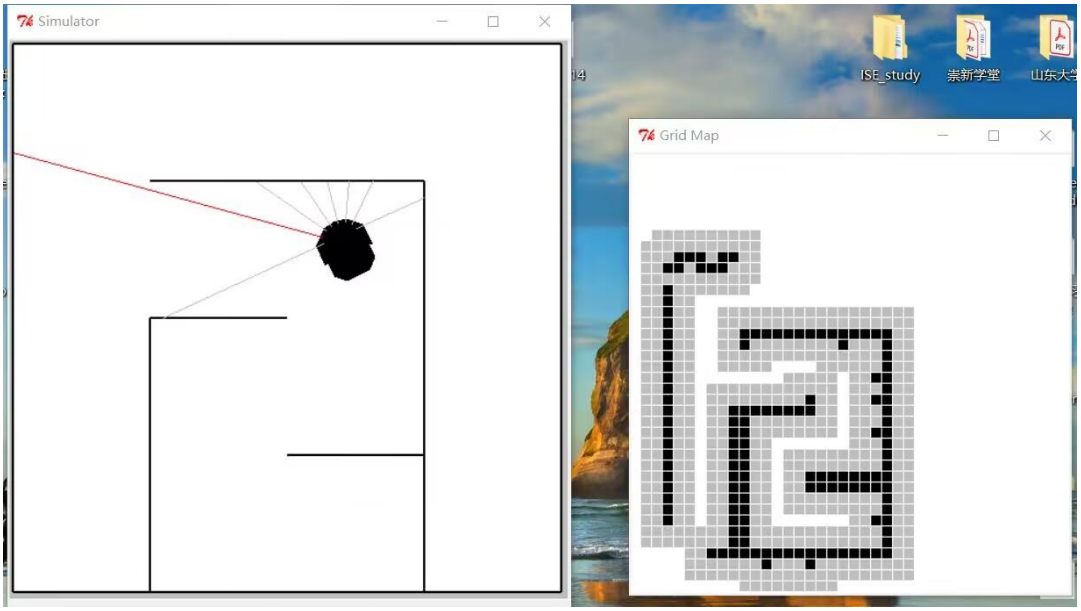

Figure 9 action of the car under medium noise

Step6 .Ameliorate MapMaker class

According to the lab handout, we can use util.lineIndices(start, end) to remove the incorrectly generated black barriers.In the process of improvement, we found that it is also necessary to limit the sonar that exceeds the effective value of sonar detection.We set the data whose sonar distance is greater than the maximum effective value as the maximum effective value, so that the generated map will not be greatly damaged, the code is as follows:

```
def getNextValues(self, state, inp):
    sonar=inp.sonars
    pose=inp.odometry
    for i in range(len(sonarDist.sonarPoses)):
        if sonar[i]< sonarDist.sonarMax:
            black=state.pointToIndices(sonarDist.sonarHit(sonar[i],sonarDist.sonarPoses[i],pose))#hits location
            state.setCell(black)
            for j in util.lineIndices(state.pointToIndices(pose.transformPoint(sonarDist.sonarPoses[i].point())),black)[:-1]:#clear
                state.clearCell(j)
        else :
            sonar[i]=sonarDist.sonarMax
            black=state.pointToIndices(sonarDist.sonarHit(sonar[i],sonarDist.sonarPoses[i],pose))
            state.setCell(black)
            for j in util.lineIndices(state.pointToIndices(pose.transformPoint(sonarDist.sonarPoses[i].point())),black)[:-1]:
                state.clearCell(j)
    return(state,state)
```

Figure 10 improvement code

Step7 and Check Yourself 3. Test new MapMaker in Idle

run in Idel:

testMapMakerClear(testClearData)

result:

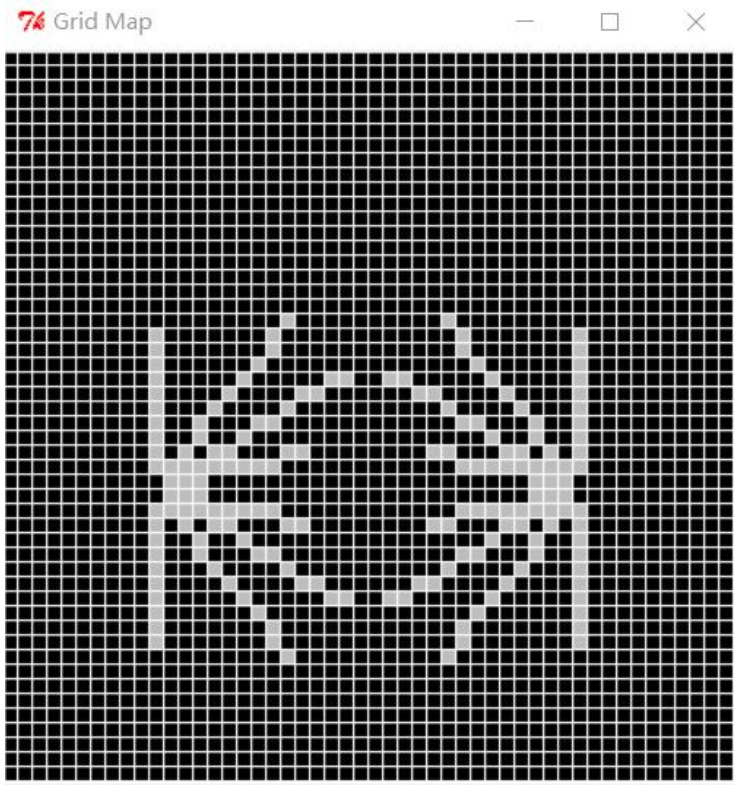

Figure 11 result map

We can see that this clears all the black bodies within the effective range of the sonar and retains the black obstacles at the end of the sonar.

Step 8 and Checkoff 2.*Run in soar again*

no noise：

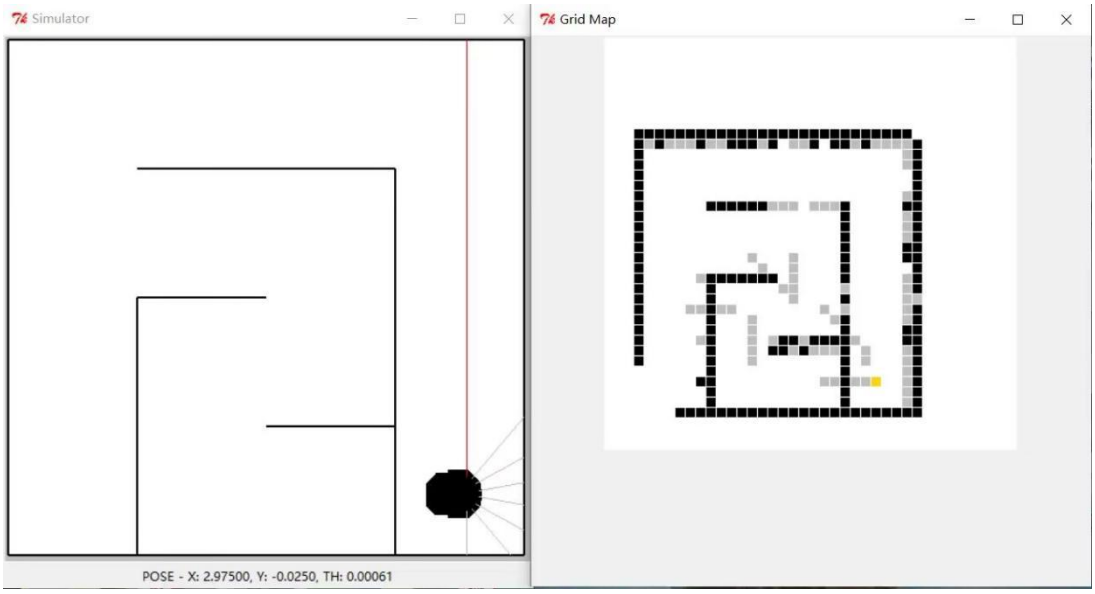

Figure 12 no noise

medium noise:

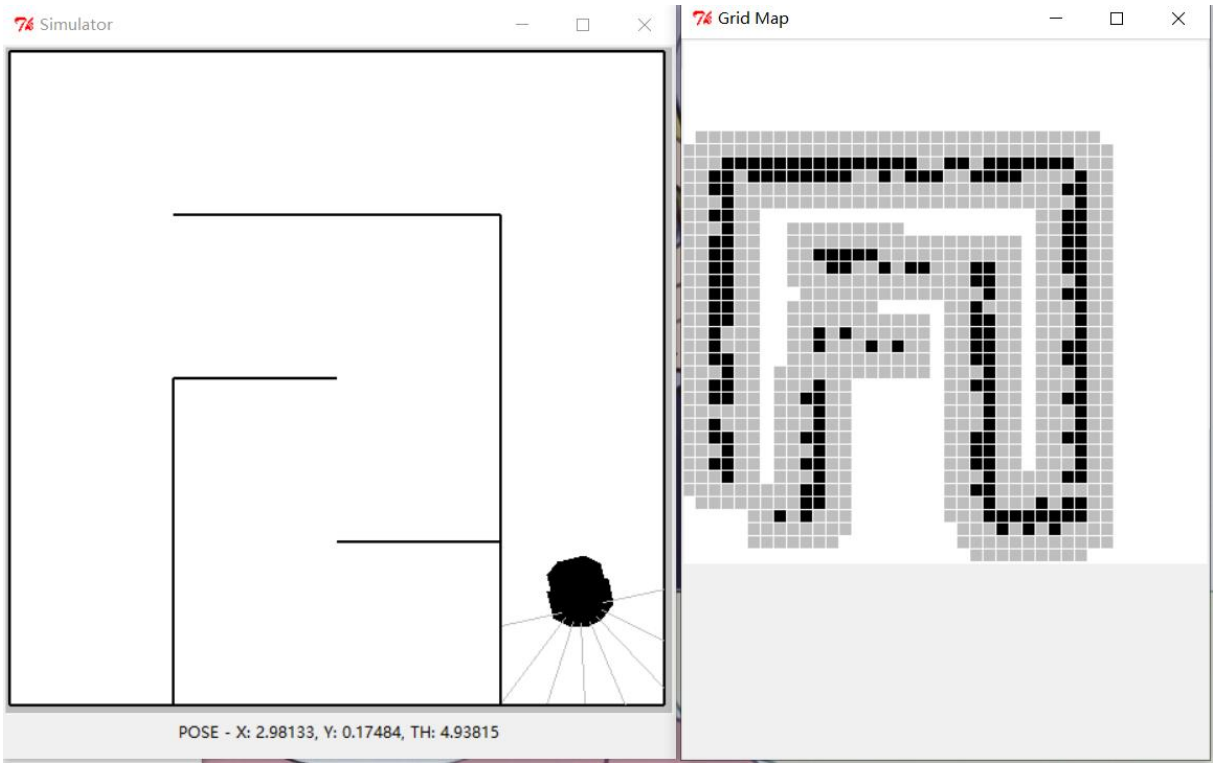

Figure 13 medium noise（Multiple test results, there is uncertainty）

We can see that although the action effect of the car is not good in the

medium noise environment, and it does not run out of the complete map, it can be seen that the car system is constantly correcting the map and modifying the path to achieve the goal.

Checkoff Yourself 4.

What do you think the likelihood is that we observe a cell to be free when it is really occupied?

$$hitP1 = 0.3$$

$$freeP1 = 0.7$$

That we observe it as a hit when it is really not occupied?

$$hitP2 = 0.75$$

$$freeP2 = 0.25$$

A transition model is a conditional probability distribution expressed as a distribution that takes actions as input and returns a process and the resulting state (discrete robot position). We consider that the input state is the same as the result state, that is, set to 1.

Step9. Create an instance of ssm.StochasticSM

We start by creating an observation model oGivenS(s) and entering the probability distributions for the two observations as follows：

```
def oGivenS(state):
    if state == 'empty':
        return dist.DDist({'hit': hitP1, 'free': freeP1})
    return dist.DDist({'hit': hitP2, 'free': freeP2})
```

Figure 14 observation model

Then create uGivenAS(a) with the following code:

```
def uGivenAS(a):
    return lambda s: dist.DDist({s: 1.0})
```

Figure 15 transformation model

combined it:

```
cellSSM = ssm.StochasticSM(dist.DDist({'occ': initPOcc, 'empty': 1 - initPOcc}), uGivenAS, oGivenS)
```

Figure 16 cellSSM

Step10.Test grid cell model

1.What is its final degree of belief that the cell is occupied if you give it this input data? (Why are the Nones here?)

mostlyHits = [('hit', None), ('hit', None), ('hit', None), ('free', None)]

```
[DDist(occ: 0.762712, empty: 0.237288), DDist(occ: 0.960171, empty: 0.039829), DDist(occ: 0.994
500, empty: 0.005500), DDist(occ: 0.980478, empty: 0.019522)]
```

Figure 17 mostlyHits result

2.How about if you give it this input data?

mostlyFree = [('free', None), ('free', None), ('free', None), ('hit', None)]

```
>>>
[DDist(occ: 0.106383, empty: 0.893617), DDist(occ: 0.032010, empty: 0.967990), DDist(occ: 0.0091
02, empty: 0.990898), DDist(occ: 0.064453, empty: 0.935547)]
>>>
```

Figure 18 mostlyFree result

step11.think through a strategy

We will emulate the DynamicGridMap class to achieve this goal

```
class DynamicGridMap(gridMap.GridMap):
    def makeStartingGrid(self):
        return util.make2DArray(self.xN, self.yN, False)
    def squareColor(self, (xIndex, yIndex)):
        if self.occupied((xIndex, yIndex)): return 'black'
        else: return 'white'
    def setCell(self, (xIndex, yIndex)):
        self.grid[xIndex][yIndex] = True
        self.drawSquare((xIndex, yIndex))
    def clearCell(self, (xIndex, yIndex)):
        self.grid[xIndex][yIndex] = False
        self.drawSquare((xIndex, yIndex))
    def occupied(self, (xIndex, yIndex)):
        return self.grid[xIndex][yIndex]
```

?

Figure 19 DynamicGridMap class

Step12. Wk.14.2.3 Solve this tutor problem on making collections of object instances.

part1: Enter the required code and answer the questions

```
def lotsOfClass(n, v):
    one = MyClass(v)
    result = []
    for i in range(n):
        result.append(one)
    return result

class10 = lotsOfClass(10, 'oh')

class10[0].v = 'no'
```

1. What is the value of `class10[0].v`:
2. What is the value of `class10[3].v`:

Figure 20 code and questions

result:

```
>>>
>>> class10[0].v
'no'
>>> class10[3].v
'no'
>>>
```

Figure 21 part1 answer

part2:change the code:

```
def lotsOfClass(n, v):
    result = []
    for i in range(n):
        one = MyClass(v)
        result.append(one)
    return result
```

Figure 22 part2 code

result:

```
>>> class10[0].v
'no'
>>> class10[3].v
'oh'
>>>
```

Figure 23 part2 answer

part3: Use util.makeVectorFill()

code:

```
def Class1(v):
    one = MyClass(v)
    return one
def lotsOfClass(n, v):
    result = util.makeVectorFill(n,lambda x:Class1(v))
    return result
```

Figure 24 part3 code

result:

```
>>> class10[0].v
'no'
>>> class10[3].v
'oh'
>>>
```

Figure 25 part3 answer

Step13 and Step14. implementing the BayesGridMap class

The code is modeled after the DynamicGridMap class:

```
class BayesGridMap(dynamicGridMap.DynamicGridMap):

    def squareColor(self, (xIndex, yIndex)):
        p = self.occProb((xIndex, yIndex))
        if self.robotCanOccupy((xIndex,yIndex)):
            return colors.probToMapColor(p, colors.greenHue)
        elif self.occupied((xIndex, yIndex)):
            return 'black'
        else:
            return 'red'
    def occProb(self, (xIndex, yIndex)):
        sm = self.grid[xIndex][yIndex]#每个点都是状态机
        return sm.state.prob('occ') #提取occ的概率
    def makeStartingGrid(self):
        def makeEstimator(ix, iy):
            m = seFast.StateEstimator(cellSSM)
            m.start()
            return m
        return util.make2DArrayFill(self.xN, self.yN, makeEstimator)

    def setCell(self, (xIndex, yIndex)):
        sm = self.grid[xIndex][yIndex]
        sm.step(('hit', None))
        self.drawSquare((xIndex, yIndex))
        pass
    def clearCell(self, (xIndex, yIndex)):
        sm = self.grid[xIndex][yIndex]
        sm.step(('free', None))
        self.drawSquare((xIndex, yIndex))
        pass
    def occupied(self, (xIndex, yIndex)):
        Pr_occ = self.occProb((xIndex, yIndex))
        return Pr_occ > occ_max
```

Figure 26 BayesGridMap class

Step15 and Check Yourself 5. Test code in Idle

Changing MapMaker to use bayesMap.BayesGridMap instead of dynamicGridMap.DynamicGridMap.

Running mapMakerSkeleton.py in Idle, and then typing in the shell:

testMapMakerN(1, testData)

result：

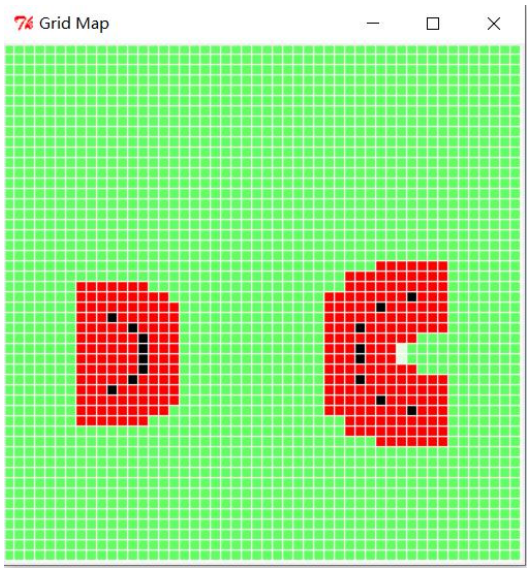

Figure 27 updata one time（The right center is aqua）

Running mapMakerSkeleton.py in Idle, and then typing in the shell:

testMapMakerN(2, testData)

result：

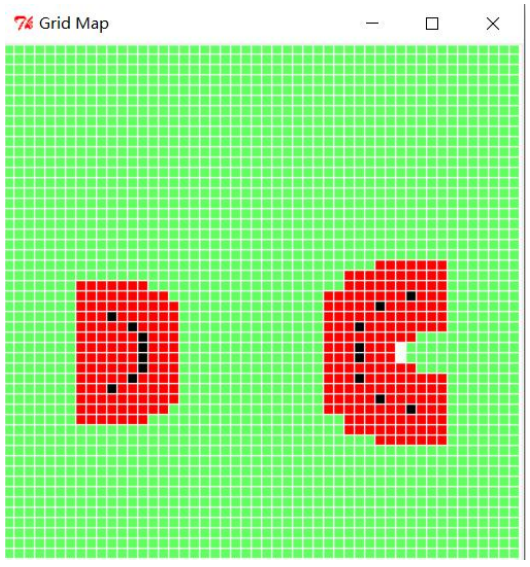

Figure 28 updata two times（The right center is white）

Running mapMakerSkeleton.py in Idle, and then typing in the shell:

testMapMakerN(1, testClearData) and testMapMakerN(2, testClearData)

result：

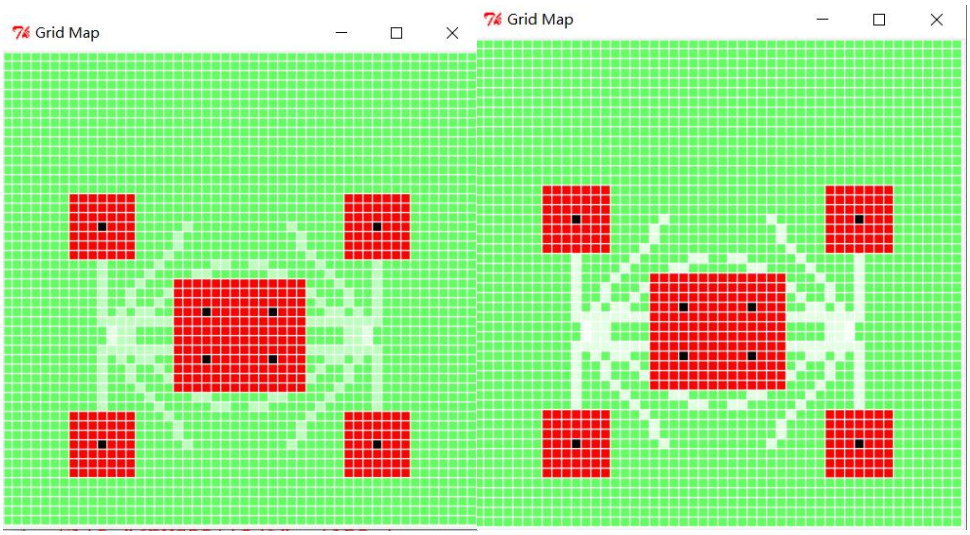

Figure 29 updata one time and two times

Step16 and Checkoff 3.Test code in soar

Medium noise：

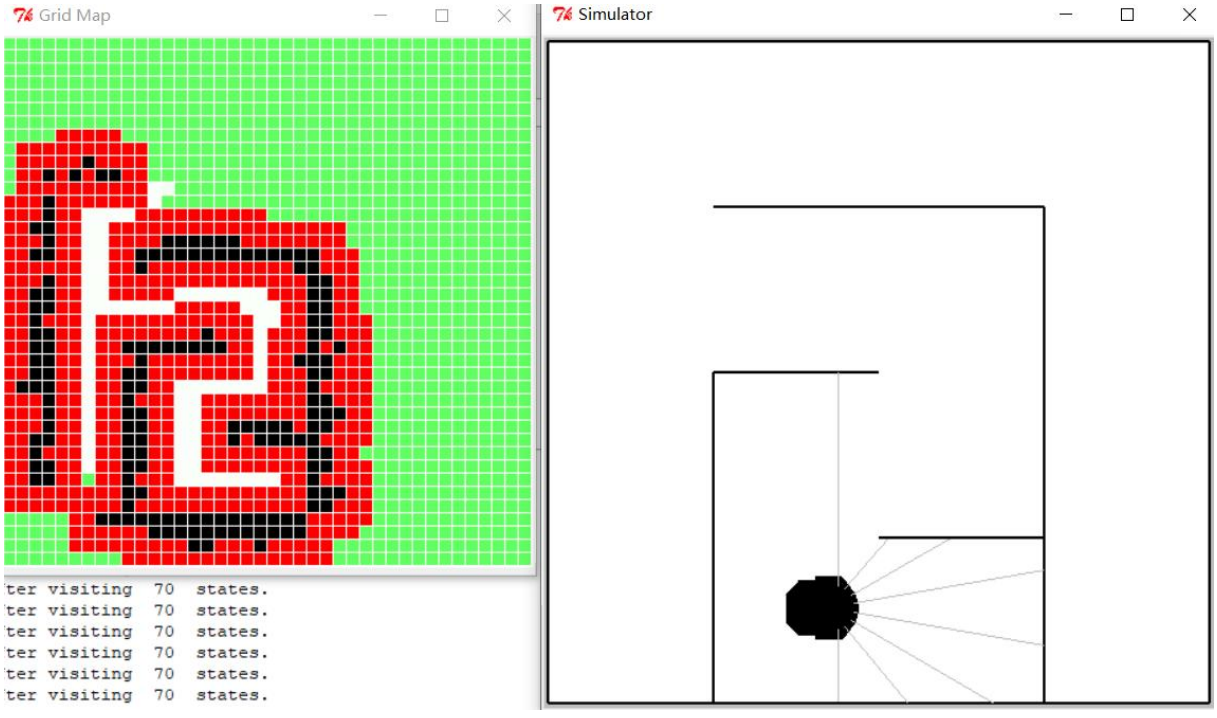

Figure 30 map under medium noise(Probabilistic success)

Big noise:

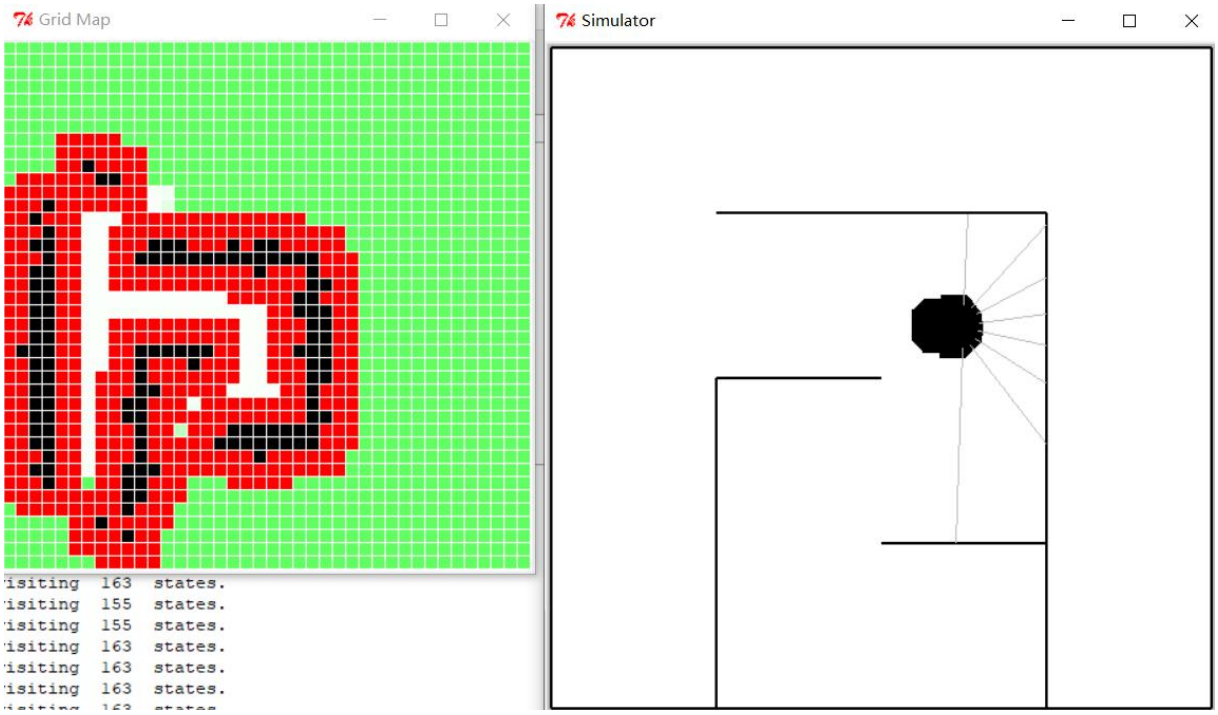

Figure 31 map under big noise(Probabilistic success)

Step 17 and Check Yourself 6.Run through raceWorld in soar

When not optimized:

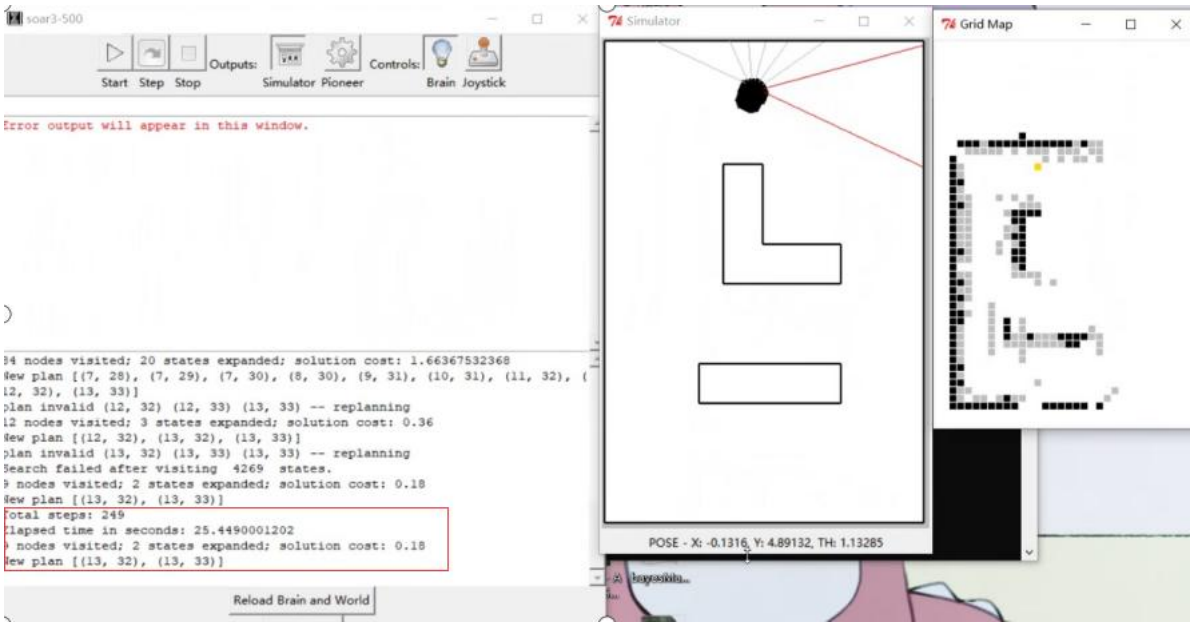

Figure 32 race in soar(not optimized)

total steps:249

Elapsed time in seconds:25.449 s

Step 18 and Check Yourself 7. Implement some improvements

We first choose method 1 to achieve the promotion:

1. Plan with the original set of actions, but then post-process the plan to make it more efficient. If the plan asks the robot to make several moves along a straight line, you can safely remove the intermediate subgoals, until the location where the robot finally has to turn.

To determine a line according to three points, we take the adjacent three points in the planned path to determine whether they are in a straight line. If on a line, we take the middle point and add it to the new list,Finally, delete the points saved in the new list from the planned path.

code:

```
if plan:
    # The call to the planner succeeded;  extract the list
    # of subgoals
    state = [s[:2] for (a, s) in plan]
    delt=[]
    for i in range(len(state)-2):
        (x1,y1)=state[i]
        (x3,y3)=state[i+2]
        if x1==x3 or y1==y3:
            delt.append(state[i+1])
    for j in delt:
        state.remove(j)
    print 'New plan', state
    # Draw the plan
    map.drawPath(state)
else:
    # The call to the plan failed
    # Just show the start and goal indices, for debugging
    map.drawPath([currentIndices, goalIndices])
    state = None
```

Figure 33 code

And then modify the gain:

```
move.MoveToDynamicPoint.forwardGain = 1.5
move.MoveToDynamicPoint.rotationGain = 1.5
move.MoveToDynamicPoint.angleEps = 0.05
```

Figure 34 new gains

result:

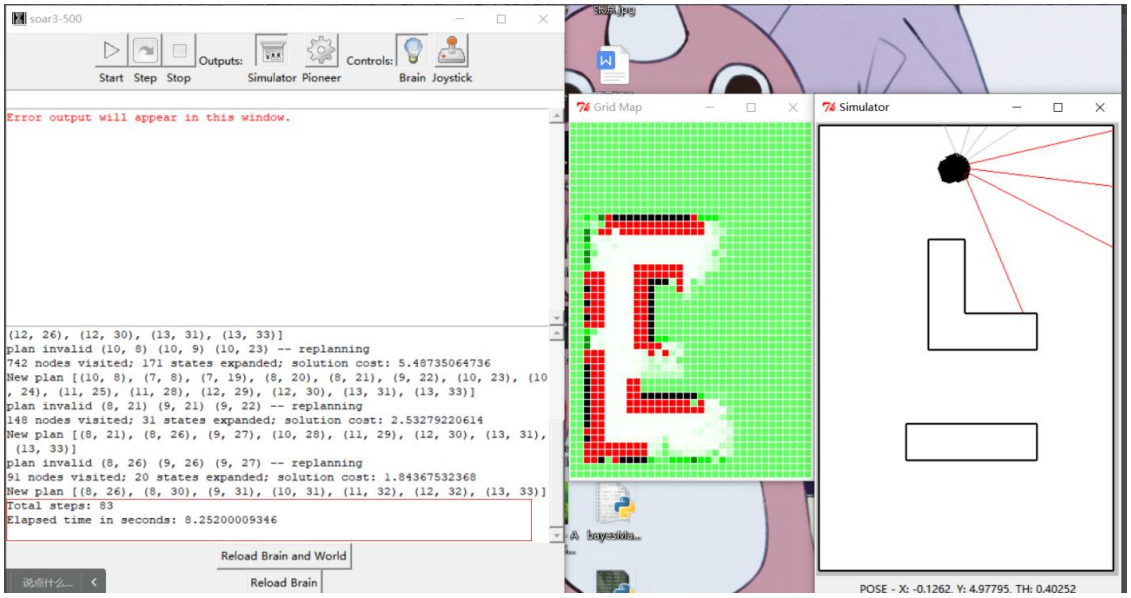

Figure 35 bayes map

bayes total steps:83

bayes Elapsed time in seconds:8.25 s

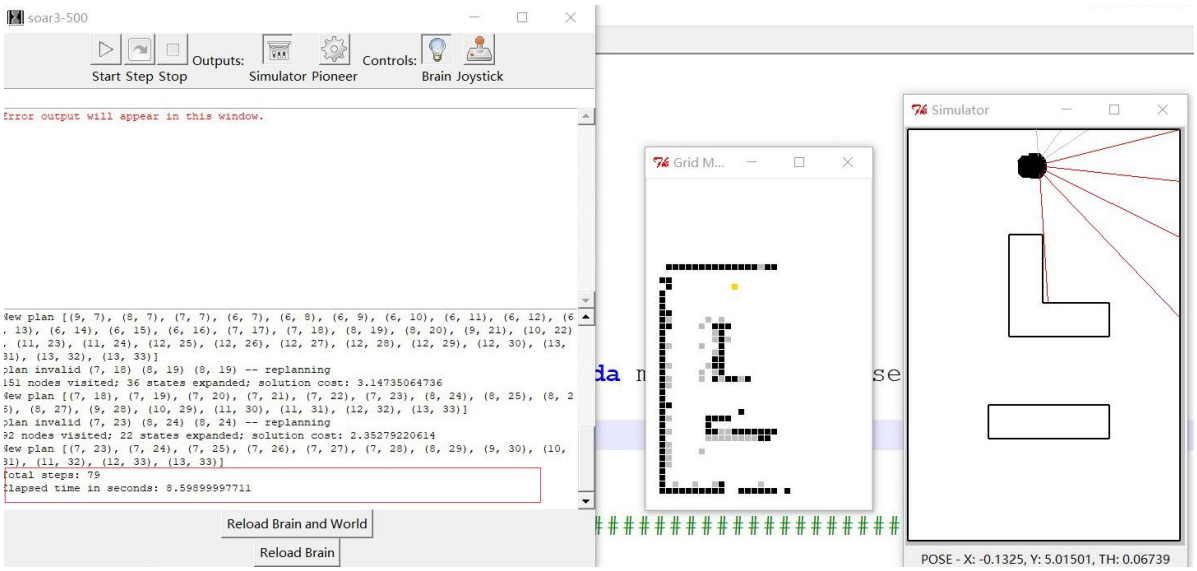

Figure 36 DynamicGridMap map

DynamicGridMap total steps:79

DynamicGridMap Elapsed time in seconds:8.598 s

Summary

1.At the beginning of the problem, the black square, white square, gray square understanding was a little confused, but by trying the setcell() function eventually completely understood

2.In step7, the original code did not take into account the case of exceeding the Effective value of sonar, resulting in the following error:

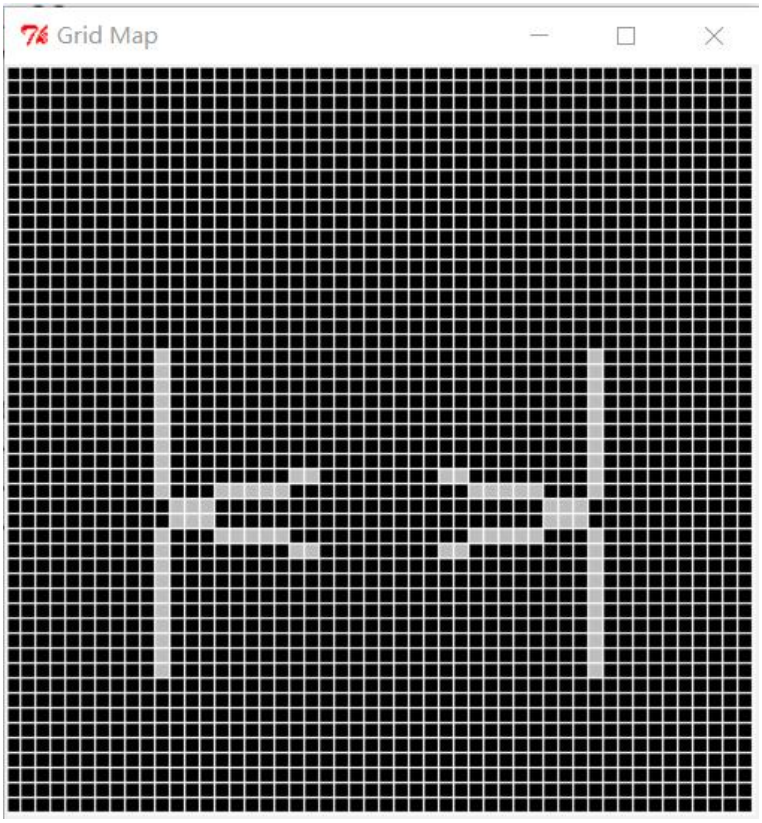

Figure 37 mistake map1

However, the sonar value exceeding the effective value is not limited, which may damage the map, as shown in the figure:

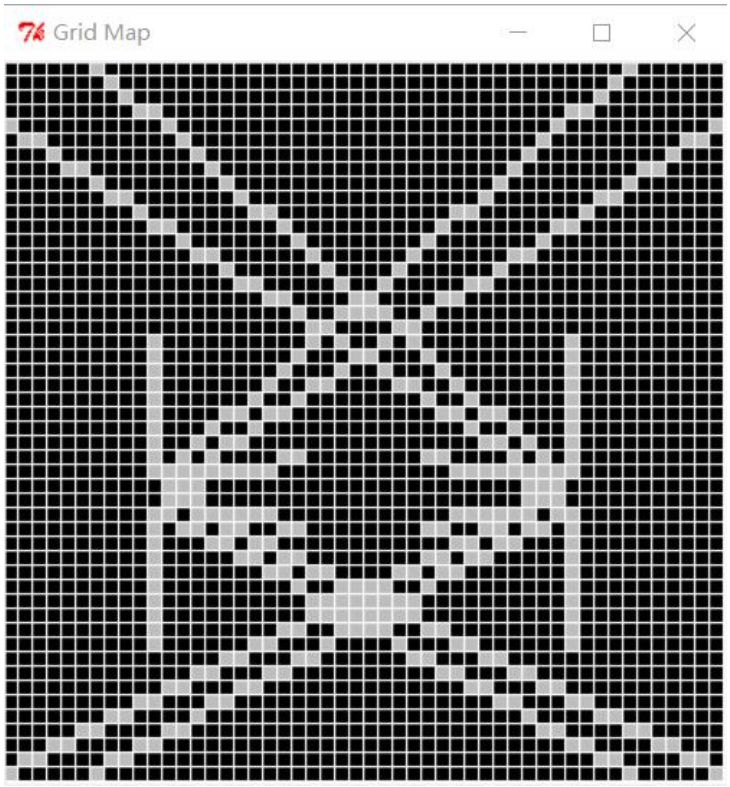

Figure 38 mistake map2

Finally, the correct result was obtained after several modifications:

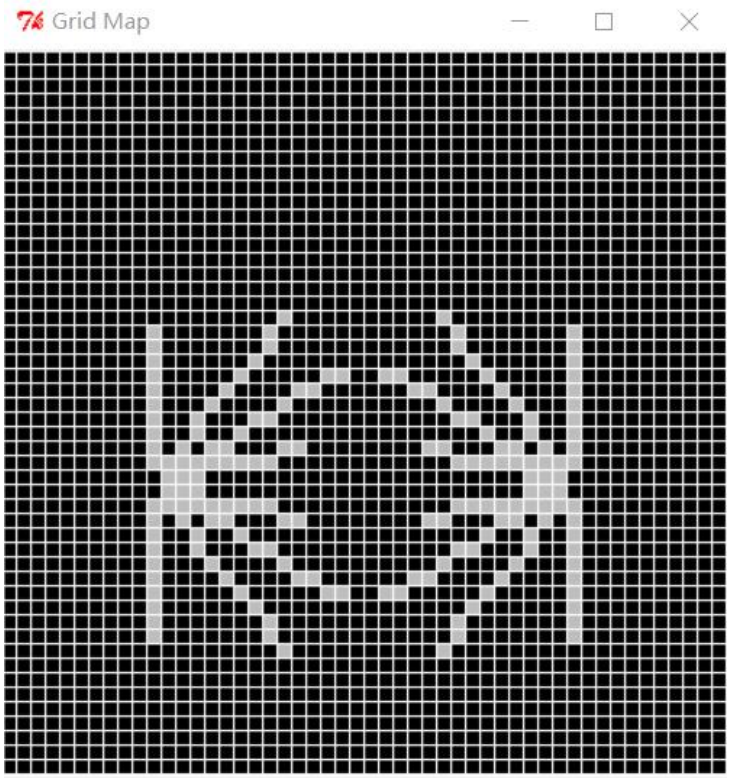

Figure 39 right map

3.Good results have been obtained in many experiments of racing

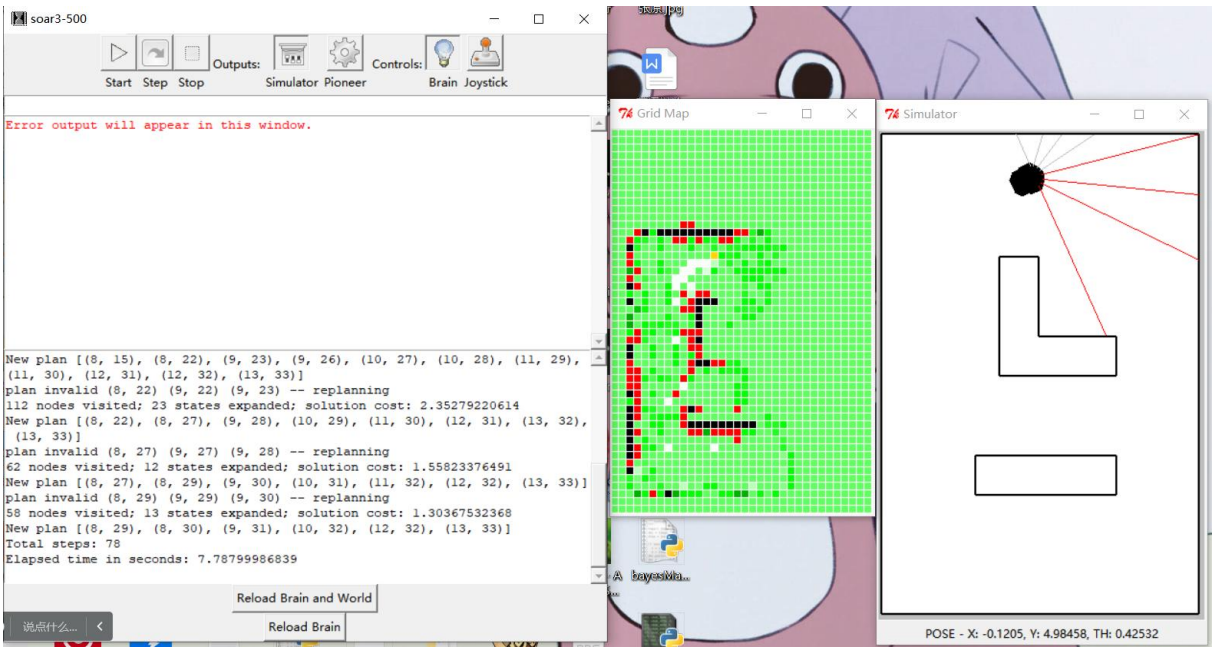

Figure 40 race result

bayes total steps:78

bayes Elapsed time in seconds:7.87 s

4.Through this course, we really realized that when doing engineering, you need to constantly identify problems, solve problems, and realize how important P(primitives)C(combination)A(abstraction)P (patterns)is.

Finally, I would like to thank Mr. Fang for his guidance, senior Wang Genglin for his help, and the team members for their unremitting efforts. We may not have been the best group, but we really did the best we could with all the experiments.