HW5: Car Tracking

110550093 蔡師睿

Part I. Implementation

Part 1.

```
# BEGIN_YOUR_CODE
"""
1. Compute the distance between your position and the observation.
2. Use util.pdf() to compute the probability density for a Gaussian where its mean is the computed distance.
3. Set the probability of each tile to be its pdf times the probability of that tile.
4. Normalize self.belief to make the all probabilities sum to 1.
"""
for row in range(self.belief.getNumRows()):
    for col in range(self.belief.getNumCols()):
        dist = ((util.colToX(col) - agentX) ** 2 + (util.rowToY(row) - agentY) ** 2) ** 0.5
        self.belief.setProb(
            row,
            col,
            util.pdf(dist, Const.SONAR_STD, observedDist) * self.belief.getProb(row, col),
        )
self.belief.normalize()
# END_YOUR_CODE
```

Part 2.

```
# BEGIN_YOUR_CODE
"""
1. Create a new belief which has the same row and column as self.belief.
2. For each items in self.transProb, compute the transProb times the probability of oldTile.
3. Increase the probability of newTile by the computed value.
4. Normalize the new belief and update self.belief to the new belief.
"""
new_belief = util.Belief(self.belief.getNumRows(), self.belief.getNumCols(), 0)
for (old, new), prob in self.transProb.items():
    delta = prob * self.belief.getProb(old[0], old[1])
    new_belief.addProb(new[0], new[1], delta)
new_belief.normalize()
self.belief = new_belief
# END_YOUR_CODE
```

Part 3-1.

```
# BEGIN_YOUR_CODE
"""
1. Create two dictionaries called "reweight" and "new_particles".
2. For each items in self.particles, compute the distance observation and your position.
3. Use util.pdf() to compute the probability density for a Gaussian where its mean is the distance.
4. Put the pdf times the number of particles at that grid square in "reweight" which its key is the
   grid location.
5. Resample |self.NUM_PARTICLES| times using util.weightedRandomChoice() in "new_particles".
6. Update self.particles to the new_particles.
"""
reweight = collections.defaultdict(float)
new_particles = collections.defaultdict(int)
for (row, col), num in self.particles.items():
    dist = ((util.colToX(col) - agentX) ** 2 + (util.rowToY(row) - agentY) ** 2) ** 0.5
    reweight[(row, col)] = num * util.pdf(dist, Const.SONAR_STD, observedDist)
for i in range(self.NUM_PARTICLES):
    new_particles[util.weightedRandomChoice(reweight)] += 1
self.particles = new_particles
# END_YOUR_CODE
```

Part 3-2.

```
# BEGIN_YOUR_CODE
"""
1. Create a new dictionary called "new_particles".
2. For every items in self.particles, run "num" times to update the "new_particles" using
   util.weightedRandomChoice() with the transition probabilities of the location.
3. Update self.particles to the "new_particles".
"""
new_particles = collections.defaultdict(int)
for particle, num in self.particles.items():
    for i in range(num):
        new_particles[util.weightedRandomChoice(self.transProbDict[particle])] += 1
self.particles = new_particles
# END_YOUR_CODE
```

Part II. Question Answering

Please describe problems you met and how you solved them.

1. After finishing Part 3, I ran drive.py using particleFilter, but my car looked something wrong. Then, I reread the comment and finally found that the keys in self.particles are in the form of (row, col). Therefore, I altered the order of row and col.

2. After solving the bug 1 mentioned above, I discovered that particleFilter made my car drive up and down the street before heading towards the target area. Theoretically, it's impossible. I have rechecked my code, but I don't think that there is anything wrong in my code. Finally, I decided to download HW5 again from E3, and surprisingly, the weird action of my car disappeared. Overall, the bug is really confusing.