Homework 5: Car Tracking

110550108 施柏江

Part I. Implementation :

Part 1

```python
def observe(self, agentX: int, agentY: int, observedDist: float) -> None:
    """
    We first calculate the probability of each cell being the agent's actual position
    using a distance-based probability distribution. Then, we update the belief
    probabilities and normalize them. Overall, we update the belief probabilities
    based on the observed distance from the agent's current position.
    """
    # BEGIN_YOUR_CODE
    for row in range(self.belief.numRows):
        for col in range(self.belief.numCols):
            dis = math.dist([agentX, agentY], [util.colToX(col), util.rowToY(row)])
            pdf = util.pdf(dis, Const.SONAR_STD, observedDist)
            updated_prob = self.belief.getProb(row, col) * pdf
            self.belief.setProb(row, col, updated_prob)
    self.belief.normalize()
    # END_YOUR_CODE
```

Part 2

```python
def elapseTime(self) -> None:
    if self.skipElapse: ### ONLY FOR THE GRADER TO USE IN Part 1
        return
    """
    We first calculate the contribution of each old tile to the new tiles.
    Then, we update the belief probabilities and normalize them. Finally, we
    assign the updated belief grid to 'self.belief'. Overall, we update the
    agent's belief about its location based on a transition probability matrix.
    """
    # BEGIN_YOUR_CODE
    new_belief = util.Belief(self.belief.numRows, self.belief. numCols, 0)
    for ((oldTile, newTile), transProb) in self.transProb.items():
        delta = self.belief.getProb(oldTile[0], oldTile[1]) * transProb
        new_belief.addProb(newTile[0], newTile[1], delta)
    new_belief.normalize()
    self.belief = new_belief
    # END_YOUR_CODE
```

Part 3-1

```python
def observe(self, agentX: int, agentY: int, observedDist: float) -> None:
    """
    We first calculate the re-weighted probabilities for each particle by
    multiplying the original probability with a distance-based probability
    distribution. Then, we perform resampling to select new particles from
    the re-weighted probabilities and update the set of particles.
    Overall, we update the agent's set of particles based on an observed distance.
    """
    # BEGIN_YOUR_CODE
    reWeighted = collections.defaultdict(float)
    for (row, col), num_of_particles in self.particles.items():
        dis = math.dist([agentX, agentY], [util.colToX(col), util.rowToY(row)])
        pdf = util.pdf(dis, Const.SONAR_STD, observedDist)
        reWeighted[(row, col)] = self.particles[(row, col)] * pdf

    reSampled = collections.defaultdict(int)
    for i in range(self.NUM_PARTICLES):
        particle = util.weightedRandomChoice(reWeighted)
        reSampled[particle] += 1
    self.particles = reSampled
    # END_YOUR_CODE
```

Part 3-2

```python
def elapseTime(self) -> None:
    """
    We first select new particles for each existing particle based on the
    transition probabilities. Then, we update the set of particles. Overall,
    we update the agent's set of particles based on a transition probability dictionary.
    """
    # BEGIN_YOUR_CODE
    new_particles = collections.defaultdict(int)
    for particle, num in self.particles.items():
        for i in range(num):
            weight = self.transProbDict[particle]
            new_particle = util.weightedRandomChoice(weight)
            new_particles[new_particle] += 1
    self.particles = new_particles
    # END_YOUR_CODE
```

Part II. Question answering :

This assignment was quite abstract and involved several concepts related to probability, which made it challenging to understand the specifications and the video. As a result, I spent a lot of time trying to understand the explanations and sought information through online searches and discussions with classmates. Eventually, I was able to complete the assignment successfully.