# Homework 2: Uncertainties

**Name:** Niharika Shrivastava
**Matriculation Number:** A0254355A

| Problem | Max Points | Points |
|---|---|---|
| 1 | 10 | |
| 2 | 10 | |
| 3 | 15 | |
| 4 | 20 | |
| 5 | 10 | |
| 6 | 15 | |
| 7 | 10 | |
| **Total** | **90** | |

---

1. A suitable distance sensor for:
    a. **a light-weight UAV: Ultrasonic sensor > Camera > GPS > LiDAR**
    It would be preferable to use a cheap, light-weight, and fairly accurate sensor for a light-weight UAV. This is because this UAV itself is cheaper relatively than a heavy-weight UAV and accepts less payload for extra sensor weight.

| Parameter | RGB-D Camera | Ultrasonic sensor | LiDAR | GPS |
|---|---|---|---|---|
| **Accuracy** | Rely on visual recognition and can be affected by lighting conditions, environmental factors which could affect their accuracy. Cannot infer transparent materials properly. Range of 5m. | Provides sufficient accuracy for hovering in a fixed 3-D pose (accurate upto ~0.3cm) with a distance range of 11m. | More accurate than ultrasonic sensors (~1mm). Has a range of ~60m. | May not provide sufficient accuracy for indoor or obstructed environments. |

| Parameter | RGB-D Camera | Ultrasonic sensor | LiDAR | GPS |
|---|---|---|---|---|
| **Weight** | Are light-weight and easy to install. | Light-weight design crucial for flight performance. | Much heavier than an ultrasonic sensor/camera. | Light-weight |
| **Cost** | Costlier than ultrasonic sensors or a normal RGB camera. | Replatively inexpensive compared to LIDAR. | A high-quality one is expensive. | Comparatively expensive. |

**b. A full-sized outdoor UAV: GPS > LIDAR**

It would be preferable to have high accuracy, not that much heavy-weight (since the UAV is itself heavy and it needs to be in-flight all the time, though slightly heavy than light-weight UAV sensors is acceptable), and cost-effective.

| Parameter | RGB-D Camera | Ultrasonic sensor | LiDAR | GPS |
|---|---|---|---|---|
| **Accuracy** | Rely on visual recognition and can be affected by lighting conditions, environmental factors which could affect their accuracy. Cannot infer transparent materials properly. Range of 5m. | Provides sufficient accuracy for hovering in a fixed 3-D pose (accurate upto ~0.3cm) with a distance range of 11m. | More accurate than ultrasonic sensors (~1mm). Has a range of ~60m. They also work well in diverse lighting conditions and challenging environments. | GPS with Real-Time Kinematic (RTK) capability provide high accuracy for greater altitudes. Support longer ranges than standard GPS or ultrasonic sensors. |
| **Weight** | Are light-weight and easy to install. | Light-weight design crucial for flight performance. | Heavier than an ultrasonic sensor/camera. | Light-weight |
| **Cost** | Much cheaper than LiDAR or a high-quality GPS. | Replatively inexpensive compared to LIDAR. | A high-quality one is expensive. | Comparatively expensive. |

---

2. Girl-and-Tiger:
   a.

| Time | Observation | Unnormalized P(Left Door) | Unnormalized P(Right Door) | P(Left Door) | P(Right Door) |
|---|---|---|---|---|---|
| **0** | N/A | 0.5 | 0.5 | 0.5 | 0.5 |
| **1** | TL | 0.5*0.85 = 0.425 | 0.5*0.15 = 0.075 | **0.85** | **0.15** |
| **2** | TR | 0.85*0.15 = 0.1275 | 0.15*0.85 = 0.1275 | **0.5** | **0.5** |
| **3** | TR | 0.5*0.15 = 0.075 | 0.5*0.85 = 0.425 | **0.15** | **0.85** |

b.

| Time | Observation | Before Jump | | | | After jump | |
|---|---|---|---|---|---|---|---|
| | | Unnormalized | | Normalized | | Normalized | |
| | | P(Left Door) | P(Right Door) | P(Left door) | P(Right door) | P(Left Door) | P(Right Door) |
| **0** | N/A | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 |
| **1** | TL | 0.5*0.85 = 0.425 | 0.5*0.15 = 0.075 | **0.85** | **0.15** | 0.9*0.85 + 0.1*0.15 = **0.78** | 0.9*0.15 + 0.1*0.85 = **0.22** |
| **2** | TR | 0.78*0.15 = 0.117 | 0.22*0.85 = 0.187 | **0.385** | **0.615** | 0.9*0.385 + 0.1*0.615 = **0.408** | 0.9*0.615 + 0.1*0.385 = **0.592** |
| **3** | TR | 0.408*0.15 = 0.0612 | 0.592*0.85 = 0.5032 | **0.1084** | **0.8915** | 0.9*0.1084 + 0.1*0.8915 = **0.1868** | 0.9*0.8915 + 0.1*0.1084 = **0.8132** |

---

3. Particle Filtering
    a. Probability that 2 particles represent the same locations = ½*½ + ½*½ = **½**
    Probability that 2 particles represent distinct locations = 1 - ½ = **½**

    b. After N steps, the probabilities would likely be:
    Probability (same location) = **1**, Probability (distinct location) = **0**

    c. The outcome of the previous result is undesirable as all particles now represent the same location after N steps, and thus cannot estimate the true underlying probability distribution anymore. This phenomena is called as particle collapse and defeats the purpose of using an important-sampled particle filter for state estimation.

    d. Even if we use K >> 2, it wouldn't solve the problem. The initial probability distribution will change to 1/K and the number of steps (N) to reach the probabilities in part (b) might increase, but the phenomena would still occur. This is due to the resampling step that samples particles from the previous set with its associated probability. Therefore, after N steps, it would always select the particles, all representing the same location at all times.

    e. To tackle this problem, it is essential for the robot to move around and gather observations. This would change the predicted probabilities by getting

corrected/updated via observations. This would result in greater exploration of the state space thereby creating samples representing distinct locations → avoids particle collapse.

---

4. Filtering algoritms:
    a. **Kalman filter:**
        i. It is well-suited for tracking applications where the system's sate evolves linearly over time, and the noise in the sensor measurements is Guassian and additive.
        ii. In the case of lane tracking, the lateral position of the car with respect to the center of the lane can often be modeled as a linear system with Gaussian noise (introduced from the front-view camera images for lane detection), making the Kalman filter a suitable choice.
        iii. The histogram filter would suffer from the curse of dimensionality, and the particle filter is computationally expensive for this high-dimensional state-space.

    b. **Histogram filter**
        **i.** The task to infer intentions of the human driver can be modeled such that:
        Set of available actions is [Accelerate, Decelerate].
        Set of valid states: [Give way, Not give way].
        **ii.** Therefore, since the state space and action space is small and can be discretized, the Histogram filter can be used.
        iii. For global localization in a small state space, the histogram filter is more efficient than the particle filter, through the use of dynamic programming.

    c. **Particle filter (***Monte Carlo Localization)*
        i. LIDAR readings can be noisy, and histogram filters may not be able to effectively handle the noise in the sensor measurements, as they rely on discrete bins with fixed counts or probabilities.
        ii. Particle filters can handle noisy sensor measurements more robustly, as they propagate a set of particles through the motion and measurement models, allowing for better estimation of the robot's pose even in the presence of sensor noise.
        iii. MCL combines the Particle filter approach with the concept of a known map to estimate the robot's pose (position and orientation) within the environment.
        iv. In the case of using 2D LIDAR readings for re-localization in a known map (COM1), the MCL algorithm can utilize a set of particles to represent the possible poses of the robot in the environment.
        v. The particles are propagated through the known map based on the robot's motion model and are weighted based on their consistency with

the 2D LIDAR readings. The particles are then resampled based on their weights to focus the estimation on the most likely poses of the robot.

---

5. POMDP
    a. States: {Hungry, Full}
    Actions: {Feed, Ignore}
    Observations: {Crying, Quiet}

    b. State-Transition probabilities

| | Hungry | Full |
|---|---|---|
| (Hungry, Feed) | 0 | 1 |
| (Hungry, Ignore) | 1 | 0 |
| (Full, Feed) | 0 | 1 |
| (Full, Ignore) | 0.05 | 0.95 |

    c. Observation probabilities

| | Crying | Quiet |
|---|---|---|
| Hungry | 0.8 | 0.2 |
| Full | 0.15 | 0.85 |

    d. Rewards

| | Feed | Ignore |
|---|---|---|
| Hungry | -3 | -2 |
| Full | -1 | 0 |

---

**6. Localization**
    a. POMDP is the most suitable for this task because of the following reasons:
        i. There is uncertainty in state estimation at every point and no observations, thereby eliminating MDP. It also makes the environment

partially-observable if we can create an observation transition model using the provided map.

ii. The robot has to reach the goal and avoid obstacles, therefore it is reward-oriented.

iii. POMDP formulation:

1. States: {Free, Goal, Obstacle}
2. Actions: {Left, Right, Up, Down, Stay}
3. Probabilistic observation function
4. Belief-Transition model
5. Z: There are no observations
6. Belief: { A:⅓, B: ⅓, C:⅓ }
7. Reward R(b, a, b') = sum [ R(s, a, s') * b(s) ]
    a. +1 if s = Free, s' = Free, a = {Forward, Up, Down, Left}
    b. -5 if s = Free, s' = Obstacle, a = {Forward, Up, Down, Left}
    c. +10 if s = Free, s' = Goal, a = {Forward, Up, Down, Left}
    d. +10 if s = Goal, s' = Goal, a = Stay
    e. -100 if s = Goal, s' = {Free, Obstacle}, a = {Forward, Up, Down, Left}

b. **Policy**

i. Since there are no observations, consider the grid as a Map and use it to build an observation model p(z | s, M).

ii. Assume the observation is 3 consecutive free cells in any directions including the current state (assuming the robot is spherical and an orientation doesn't make sense). Therefore, the observation model becomes:

| | | | | |
|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 |

iii. Using this, the probabilistic state-transition model becomes:

| | | | | |
|---|---|---|---|---|
| 1/4 | 0 | 0 | 0 | 1/4 |
| 1/4 | 0 | 0 | 0 | 1/4 |

| | | | | |
|---|---|---|---|---|
| 1/2 | 0 | 0 | 0 | 1/2 |
| 3/4 | 1/4 | 1/2 | 1/4 | 3/4 |
| 1/2 | 0 | 0 | 0 | 1/2 |
| 1/4 | 0 | 0 | 0 | 1/4 |
| 1/4 | 0 | 0 | 0 | 1/4 |

iv. **Combining the rewards as mentioned in part (A), observation model, belief, and state-transition model, we have everything required to formulate a POMDP. We can solve this to get the policy using any quick POMDP algorithm like QMDP where we update our belief at each timestep using our observation function. It is easy to compute since it is not high-dimensional.**

v. From the slides,

- For each action $a \in A$, calculate

$$Q(b,a) = R(b,a) + h(b')$$
$$= \sum_{s \in S} \left\{ R(s,a)b(s) + \sum_{s' \in S} V_{\text{MDP}}(s')p(s' \mid s,a)b(s) \right\}$$

uncertainty of s; uncertainty of actions; calculate b'

$$\pi_{\text{QMDP}}(b) = \arg\max_{a \in A} Q(b,a)$$

**vi. This will always result in reaching the goal with probability 1.**

---

7. **QMDP**

$R_A(a) = -2$

$R(s, a) = R_S(s) + R_A(a)$

| | | |
|---|---|---|
| 10 | 0 | 0 |
| 0 | -20 | 0 |
| 0 | 0 | 0 |

$R_s$

| | | |
|---|---|---|
| 10 | 8 | 6 |
| 8 | -20 | 4 |
| 6 | 4 | 2 |

$V_{MDP}$

| | | |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 0 | 0.3 |
| 0 | 0.1 | 0.6 |

b

a. We know,

- For each action $a \in A$, calculate

$$Q(b,a) = R(b,a) + h(b')$$
$$= \sum_{s \in S} \left\{ R(s,a)b(s) + \sum_{s' \in S} V_{\text{MDP}}(s')p(s' \mid s,a)b(s) \right\}$$

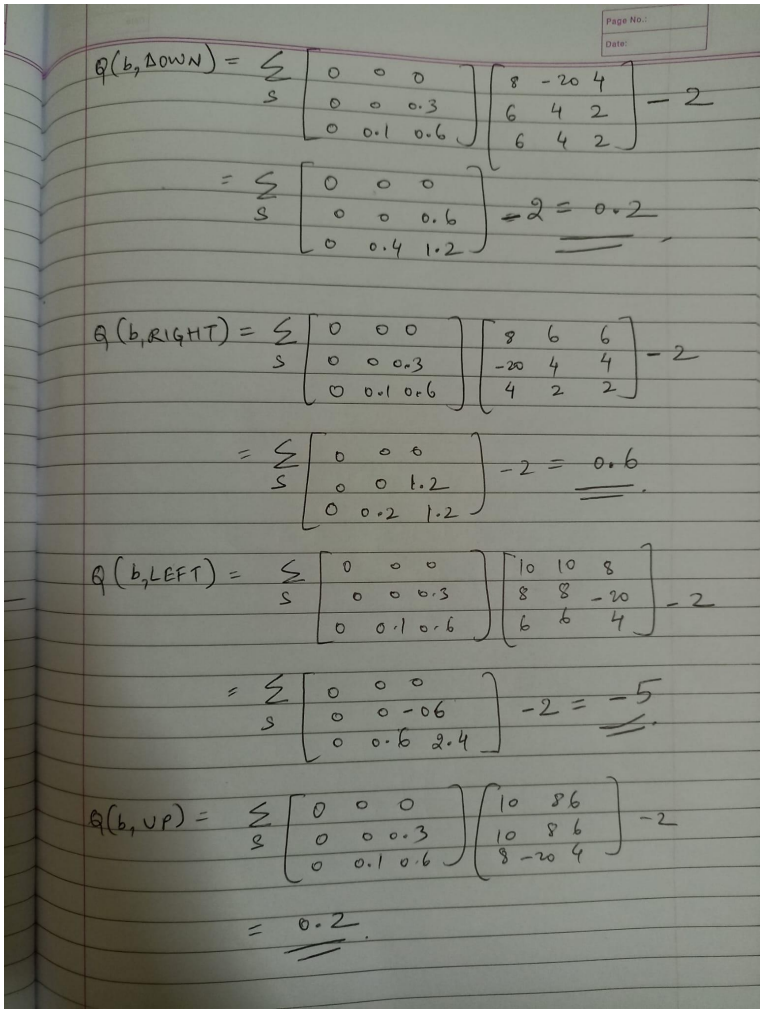
uncertainty of s; uncertainty of actions; calculate b'

Simplifying,

$Q(b, a) = \sum_s\{R_S(s) + R_A(a)\}.b + \sum_s b.\sum_{s'}V(s').p(s'| s, a)$

$Q(b, a) = -2 + \sum_s b.\sum_{s'}V(s').p(s'| s, a)$

| Q(s, UP) | Q(s, DOWN) | Q(s, LEFT) | Q(s, RIGHT) |
|---|---|---|---|
| 0.2 | 0.2 | -5 | 0.6 |



b. $$\pi_{\text{QMDP}}(b) = \arg\max_{a \in A} Q(b, a)$$

**Therefore, best action at b = RIGHT**

---