Question 4

SheepDogBot:

Properties and abilities:
The sheepdog bot should have at least one camera for vision, four wheels to run, microphones for hearing sense and one speaker to make sound. If there is a light sensor to tell the daylight or weather, a compass to tell the direction, and a GPS chip for the localization, this bot would be more powerful.

The main task of a sheepdog is to protect the sheep and shepherd them. This task can be split into several separate small tasks.

1) Observe the sheep and tell the differences between its own sheep and other people's sheep.
2) Remember the location of its home and the meadows.
3) Find the shortest path to the meadow.
4) Chase the detached sheep and make it back to the sheep flock.
5) Lead the sheep to the meadow and get back at a regular time
6) Identify the enemies.
7) Protect the sheep against enemies like wolfs or thieves by barking or attacking, or in a more robotic way, ringing the alarm and call the master.
8) Choose a proper way to shepherd, e.g., how fast should it chase after a sheep away from the flock, and how loud it should bark to notify the detached sheep without scaring the whole sheep flock, which would make the sheep flock even harder to control.
9) Find the lost sheep.

Issues and solutions:

1. How to determine if a sheep is detached from the flock.

This can be achieved by using a clustering algorithm. Take the most basic K-Nearest Neighbors (KNN) as an example. Taking the premise that the location of each sheep can be calculated from the camera. We can build a two-dimensional coordinate and mark all the sheep based on that. We can then calculate the Euclidean distances between each sheep and all other sheep and take a threshold K as the number of samples we pick. By comparing the distances, we can find the sheep with the farthest distance from other sheep, this sheep is considered as the one detached from the flock, and the bot needs to lead it back to the flock.

2. How to find a way between the home and a meadow.

   The location to shepherd might frequently be changed in the meadow as it needs time to let the grass grow again and enough for sheep to eat. This is for a sustained development target. Often the new meadow would be a new location and need to be found by the bot. The bot needs to always find the shortest path to save its energy and avoid extra trouble caused by lacking water or food while traveling. In this scenario, we need to take the prerequisite that we already have an off-line map with terrain data stored in the bot. To always get the shortest path, the bot would first need to build a model of the whole map and give each terrain different weights, e.g. a higher weight means that the path is harder to go through, we might need to give the lake a high weight, and a flatland low weight. For the place which is neighboring to the lake but still a lower weight, we can also set a negative weight to bonus this path as it is easy to go through and have a water supply. Then we have a weighted graph of the original map, and for the selected location, we can take a pathfinding algorithm, e.g., A* algorithm to find the best path.

   Moreover, if we want to lead the sheep to the meadow and make them back in a regular time, the light detection sensor and GPS chip is of vital importance. After the dog calculated the shortest path between the current location and the meadow, it should calculate the time it needs to reach the meadow and go back. In this situation, the bot needs the distance between current location and the meadow, the speed of sheep, and the current time. The speed and the location can be calculated by GPS chip, but the time estimation should be done by the bot since the meadow can be out of internet. Thus, we assume that the dog can use light senor data to tell the time. As the light sensor will only give the lighting information, we can use K-nearest neighbor to estimate the time because the size of dataset won't be very large. First, the dataset should be the lighting information together with its label(time), when the bot needs to estimate the time, the current lighting information will be inputted in the KNN model. After it get the time, the bot will roughly compute the time to shepherd, if the time is far beyond the regular time, it will choose another meadow to go.

   For the A* algorithm, we have $f(n) = g(n) + h(n)$, where the f(n) is the estimated cost from the starting point to the destination by walking through n. g(n) is the actual cost we take from the starting point to the location n. h(n) is the estimated cost of the best path from location n to the destination. By adjusting the heuristic function h(n), we can dynamically adjust the pathfinding strategy based on our need. E.g. if we already have enough water, and we want to get to the destination for food as soon as possible, then the heuristic function h(n) should be as close to the distance from the point n to the destination, this would make the A* find the shortest path. If we put the terrain weights into consideration, and we want to find a water supply during the travel, by changing the heuristic, we can find a path that is more likely to have a lake connected to the path and will not make the traveling distance too long.

3. How to identify an enemy.

This task is to tell the differences between harmless humans/animals from threatening ones. E.g. the master of the sheep and the friends and relatives of the master are harmless, and the bot should be polite to them. The horses or cows are also considered to be harmless, but a wolf or bear is extremely dangerous to the sheep and the bot should send alarms to the master and lead the sheep to fallback. A well-trained Convolutional Neural Network(CNN) would be a possible method since we have the camera that can take images in a real-time manner. When the master first gets this bot, the bot should take pictures of the master, and label the master as its owner. When the bot meets the master and detects there are other people around, the bot should also take pictures and label them as friends or relatives. To recognize friendly animals or dangerous ones, a large number of such data are available online. Thus, the CNN in the bot can be pretrained before the master get it. After taking the pictures of master and friends or relatives, the bot should re-train the convolutional neural network to update the pretrained model. The training process would be taking the face of the human in the picture, and resize the picture to a standard size, which is the input of the convolutional neural network, and change it into a grayscale image. We can regularize the value of each pixel to make the model more compatible to different situations and easy to converge. Some image enhancement mechanisms can be taken to generate more training data, e.g. generate more similar images with different brightness, contrast, randomly added noises, etc. For the labels, the people appeared around the master should be labeled as 'harmless human', which should be processed as one class using a one-hot key vector form. The wolves and bears should be labeled as 'dangerous beast' as another label, or maybe we can take it as the label wolf and label bear, this is flexible based on different configuration. The structure of the CNN may varies but mainly is made up of the input layer, convolutional layer, pooling layer, dropout layer, fully connected layer, and the output layer. The activation function can be chosen as Rectified Linear Unit(ReLU) for its outstanding performance in the convolutional computation and non-linearity. The optimizer can be an Adam Optimizer, which is based on the theory of Gradient Descent, but the difference is that Adam Optimizer can automatically adjust its learning rate, which is faster and easier to get an ideal result with less human interference, which means the hyper parameter setting can be more fexible. A well-trained CNN model can classify any creatures appeared in the camera, and the prediction result can let the bot know if this is a dangerous one or a harmless one.

4. Find a lost sheep.

   During the shepherd, some sheep might get lost if it's not in the visual range of the bot's camera and run away. To find such a lost sheep, the bot can trace the location history of the day and get the route of the sheep flock. Combining the Bayesian theory, the bot can make a prediction that this sheep is most likely to get lost in which location, and which direction has the highest possibility to run towards to. This can be achieved by giving the certain location a probability of losing a sheep. E.g. if the bot knows in some location, the camera has some blind points, then this place is more likely to lost a sheep. Also, a place with food and water is more attractive to the sheep than a place with rock or mud, this can also be taken as a condition. The bot can first calculate all the nearby places around that day's shepherd route and go to the places that are more likely to lose the sheep and search for it. If it has been to all the places nearby but still failed to find the sheep, then it can calculate the most possible route using conditional probability combining the terrain information we have from the map. Before we start the searching, the bot can also calculate the shortest path that goes through all the places we need to search to save time and energy using algorithms like Dijkstra or Breadth First Search.

5. How fast should a bot chase after a sheep and how loud should it bark.

   For a new bot without any experience, it will be difficult to know how to push a sheep back to the flock without frightening other sheep and make the situation out of control. We can use the Markov Decision Process to help the bot learn an optimal policy to shepherd such a sheep flock. At the very beginning, the bot will have to try several times with different chasing speeds and barking volumes to get the actual feedback from the sheep. We can set the reward of doing such a shepherd action by observing the reaction of sheep, e.g. if the chased sheep is running fast and other sheep are still clustered and calm, we can give this action a high reward. If the sheep are frightened and fleeing towards all directions, this action will be given a negative reward. By trying different combinations of chasing speed and barking volume, we can finally build a status-action graph based on the results and use methods like value iteration to get the optimal policy and the optimal utility of each status. This optimal policy can help the bot know under which conditions, taking what action can help control the sheep flock in the most effective way. The Markov Decision Process will help the bot become a experienced SheepDogBot, also the experience of the bot can be used to update the Markov Decision Process, which will make the bot know the habits of sheep better as the time flies.

Bonus:

i. Having completed the course, are there any projects you would revisit and improve in light of material you learned later? Be thorough.

I would like to go back to the mine sweeper project and improve it. Basically, in the mine sweeper project, we deal with the base benchmark using a logical inference way to do the searching part. When we are searching the grids in the map, we only look the eight grids around the current grids. Although we use Bayesian network to improve the searching process, the basic idea didn’t change there. We still look at the 8 grids around the current grid and assign it a weight to imply how possible it is to be a mine. But now, I think this project can be considered in a much more complex way, such like: in the searching process, we can dynamically enlarge the area we are looking at to find the mine, not just look at 8 grid every time, this can be a huge improvement to improve the accuracy of finding mines as we enlarge the knowledge base to another level.

ii. What did my dog dress as for Halloween?

Based on my knowledge base on DC universe, she dressed like superman/superwomen.

iii. Draw a picture of my dog in her Halloween costume.

S