INFO8006: Project 2 - Report

Romain LAMBERMONT - s190931
Arthur LOUIS - s191230

December 2, 2021

1 Bayes filter

a. The `_get_sensor_model` function computes the distribution $\mathbf{P}(\mathbf{e}_t|\mathbf{X}_t)$ for each $\mathbf{X}$ which means every possible position on the grid. For each cell, we suppose that the ghost is on it. We than compute the probability to get the evidence measure received.
After an analysis of the `_get_evidence` function, we can say that the noise over the real Manhattan distance between Pacman and the ghost comes from a binomial distribution with $n = \frac{SV}{p*(1-p)}$ where SV is the sensor variance and $p = 0.5$ ($n = 4 * SV$). To center the noise, a term $n * p$ is subtracted to the result of the binomial.

b. The `_get_transition_model` function computes the distribution $\mathbf{P}(\mathbf{X}_t|\mathbf{x}_{t-1})$. It is the probability for a ghost to go on a cell knowing the cell where it was previously. We know there is $P = 4 - N$ legal moves where N is the number of walls in the ghost's neighbor. We have to considerate 2 kinds of legal moves : one's that drives ghost away from Pacman and one's for the others legal moves. In function of type of legal move, the probability will be impacted : it will be k times more probable for the ghost to go in a cell that drives it away from Pacman where :

- $k = 1$ for the confused ghost
- $k = 2$ for the afraid ghost
- $k = 8$ for the scared ghost

To resume :

- let $A =$ # legal moves that drives ghost away
- let $L =$ # legal moves

We have for a legal move that drives the ghost away from Pacman : $\mathbf{P}(\mathbf{X}_t|\mathbf{x}_{t-1}) = \frac{k}{(k*A)+(L-A)}$
and for a legal move that doesn't drive the ghost away from Pacman : $\mathbf{P}(\mathbf{X}_t|\mathbf{x}_{t-1}) = \frac{1}{(k*A)+(L-A)}$
We have to compute this matrix supposing ghost in each cell. That's why the result is a 4D-matrix or in an other point of view : a matrix of matrices.

2 Implementation

a. ***See bayesfilter.py***

3 Experiment

a. To summarize Pacman's belief, a good measure would be the maximum belief in all possible positions. To compute this measure, we need to take the mean of the sum of each maximum belief relative to each ghost. By considering n ghosts in the game :

$$\text{maxBelief} = \frac{1}{n}\sum_{i=1}^{n} \max\ \mathbf{P}(\mathbf{X}_{i,t}|\mathbf{e}_{1:t})$$

By looking into the belief matrices, we notice that when Pacman is more confident of the ghost's next position, the belief is higher but when he hesitates between different spots, the probability is shared with all squares, thus the belief is lower.

b. To summarize the belief's quality, a good measure would be the Manhattan distance between the real position of the ghost and the spot relative to maxBelief. In our case, a good belief should have a low quality. Indeed, if we can predict the ghost's position correctly the Manhattan distance will converge to 0. By considering n ghosts in the game, with ghostPos(i) the position of the ghost i and maxBeliefPos(i) the case with most probability = arg max $\mathbf{P}(\mathbf{X}_{i,t}|\mathbf{e}_{1:t})$:

$$\text{beliefQuality} = \frac{1}{n}\sum_{i=1}^{n} \texttt{manhattanDistance}\ (\text{ghostPos(i)}, \text{maxBeliefPos(i)})$$

c. Thanks to the function `_record_metrics` and a script, we run our implementation 100 times for each combination of layout/ghost and for each of the 100 calls, we wait 100 time steps an we simply observe the behaviour of the ghost relatove to the belief computed by our filter. For the belief measures we find theses graphs :

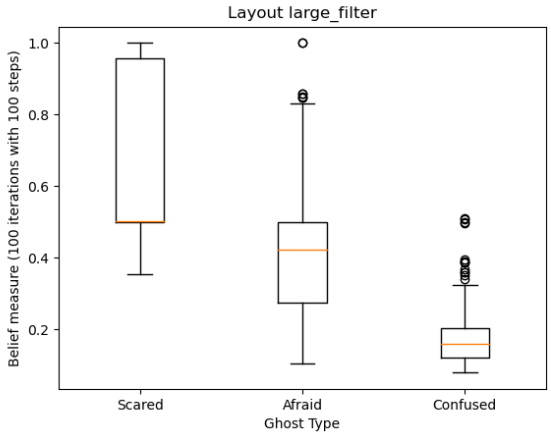

Figure 1: Measure of Belief in large_filter

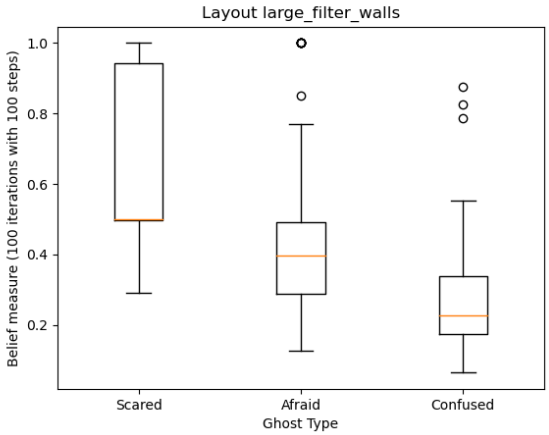

Figure 2: Measure of Belief in large_filter_walls

As expected, the belief being a probability, its value $\in [0; 1]$. This probability represents the certitude, so the higher its value, the higher the confidence. The difference between ghosts is clear. Indeed, for the less predictable ghosts (i.e. afraid and confused), our belief measure is lower, the confused ghost having logically the worst belief, his movements being completely random. For the afraid ghost, the most fearful one, his movements being much more predictable, the belief is much higher.

Then, for the belief's quality measures, we find these graphs :

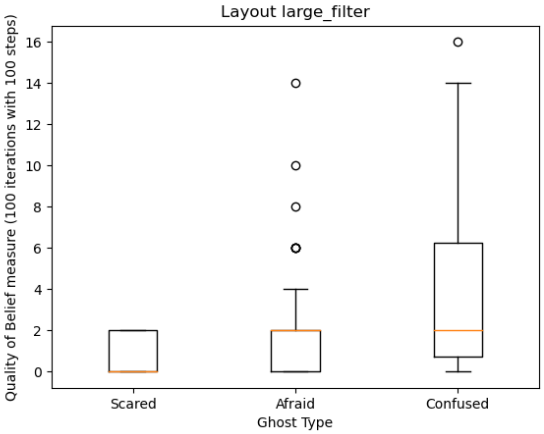

Figure 3: Belief's quality in large_filter

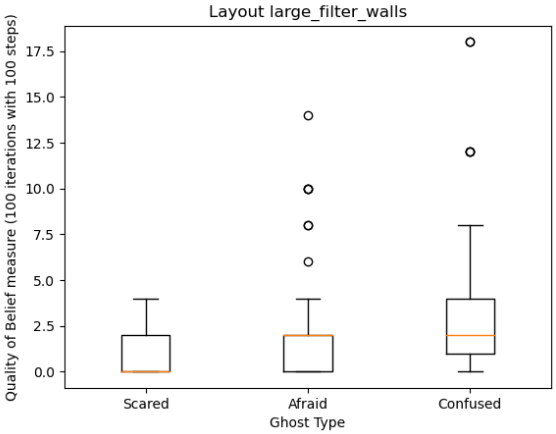

Figure 4: Belief's quality in large_filter_walls

Here, the quality measure $\in [0; \text{grid.height} + \text{grid.width}]$

d. When we need to predict the ghost's behaviour, the better the information we have, the better the predictions will be. It's easy to use the fact that some ghost tend to avoid Pacman, thus making the movements driving the ghost away from Pacman more probable. The most fearful the ghosts are, the most their movements are predictable. We can see this on the graphs. Concerning the layouts, the most walls there are in a layout, the most predictable the movements will be. Indeed, in a layout with walls, there are less legal moves for the ghosts, thus making the prediction easier.

e. The sensor variance has an effect on the belief. Indeed, the noise value is computed using a binomial distribution with parameters :

- $p = \frac{1}{2}$
- $n = \frac{\text{SV}}{p*(1-p)} = 4 * \text{SV}$, with SV being the sensor variance

If SV increases, n increases and the noise can take more values. The probabilities will spread around the different ghosts and as a result, the belief state will be worse. On the contrary, if SV decreases, n decreases, making the probabilities more focused on certain spots, resulting in a better belief state.

f. To implement our agent we try to reach the square with the maximum belief relative to a ghost (in the order of their index in the belief_states matrix, we focus on a ghost to avoid making our agent hesitate between two ghosts).
To find the shortest, we use the A* algorithm (best path finder algorithm at our disposal) but we need to recompute it at each step of the simulation because the ghosts move (relative to Pacman new position) and the belief changes at each step.

g. ***See pacmanagent.py***