Team ID: y.yang

Date of Application: Nov 1st, 2019

Link to Source Code: <https://github.com/Yang-Yuan/AnimalAI-Olympics/tree/master/agents>

# Handcrafted Animal AI Agents

## Introduction

In our implementation, we deliberately avoided using any machine learning algorithm in order to focus on exploring what faculties and prior knowledge are necessary for an animal-like agent to survive. This also serves as a good starting point to study how these faculties and prior knowledge can be acquired by agents. By comparing two agents[[1]](#footnote-1) with and without some features, we found that memory module, object-permanent representation and assumption ability together made a difference in the agent’s performance in the Animal AI competition. In this summary, we will first discuss these three features, and then complete our discussion with a brief description of our source code.

## Game-Changing Features

Our first agent (A1), as a reflex agent, was minimally designed. What it did was to simply keep rotating until any food object comes up and then go after it. When no food appeared in view, it did nothing but rotating. In contrast, our second agent (A2), equipped with game-changing features, made decisions based on its memory, did planning with object-permanent representation and used auxiliary assumptions when there was no obvious solutions. The followings table shows the online testing results of these two agents.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Total | C1 | C2 | C3 | C4 | C5 | C6 | C7 | C8 | C9 | C10 |
| A1 | 21.7 | 21 | 15 | 1 | 6 | 3 | 12 | 7 | 0 | 0 | 0 |
| A2 | 27 | 24 | 20 | 1 | 10 | 3 | 10 | 6 | 1 | 5 | 1 |

Table 1 Online testing results of our two agents

### Memory Module

The memory module records the visual inputs while the agent is rotating, and associates each visual input with the direction in which it is collected. As a result, the agent is able to make a comprehensive decision from the panorama in memory. This advantage might be used to explain C1 and C2 in Table 1 (since we don’t know what the real online tests are). According to the description in the competition website, C1(Food) and C2(Preferences) mainly involves food objects. With the help of memory, the A2 agent collects all the food in a more reasonable order that A1. Therefore, the scores of A2 are better than those of A1 in C1 and C2.

Memory has been widely used in many cognitive architectures. It is why we designed a memory module in our agent. However, it also poses another even more poignant question about how we used the memory: why and how do animals and humans know that keeping rotating will eventually put a panoramic view in memory? Where does this piece of knowledge come from?

### Object-Permanent Representation

Since there are not only moving objects but also obstacles and complicated landscapes in the arenas, it is not guaranteed that everything in the current visual input will continue to be present in the next one. For example, a moving food object can simply escapes from the agent by hiding behind a wall. In this case, there is no reason for the agent to quit but to assume that the object still exists somewhere not far from where it is seen last. Therefore, in our implementation, we estimated the representation of a missing target given its last position and the current visual input.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |

Figure 1 A sequence of snapshots showing how object-permanent representation guided the agent to a moving food that was blocked by the wall during chasing. The yellow dots indicate the path to the target (given by a A-star search), and the blue line indicates the furthest position that the agent can reach by going straight.

Similarly, since we don’t know what really happened in the online testing, we are not sure about whether this contributes to the improvement in performance. However, we do show that it works in our local testing. For example, in Figure 1 is a sequence of snapshots of our local testing. The agent firstly found a green food, but the food was moving toward behind the wall. Therefore, the agent lost the visual of the food as it was moving toward the food. However, the agent still represented the existence of the food with the position near to where it was last seen, continuing planning path and moving toward the position. As the agent approached the imaginary position, the food appeared again. This might not work if the target moves too far away from its last position, but the imaginary position is still a fairly good choice when the target is lost.

### Assumption Ability

Our path planning is based on the 2D plane of the visual input, which means that if the obstacle is too large to fit into the view of the agent’s camera, the agent won’t be able to come up with a path to bypass the obstacle. Generally speaking, it is very common for the agent to be incapable of solution unless it uses some assumptions to take action first and then validate these assumptions later. Therefore, the assumption ability could play a key role in general problem solving.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

Figure 2 A snapshot sequence showing how our assumption guided the agent to a valid path to the food.

In our second agent, we apply this methodology to implement the obstacle-avoiding function. The assumption we used is that, if it does not exist in the current visual input, then a valid path probably exists beyond the current visual input. For us humans, it is simply a common sense. But for the agent, it is nontrivial and indispensable to bypass a large obstacle. Technically, to impose this assumption on path planning, we deliberately set the boundary points on the four sides of each visual input to be accessible before passing it to the A-star search. As a result, if the target is not completely surrounded by obstacles, A-star search will always give us a path to the target as in Figure 2. In the first frame in Figure 2, a death zone traversed the view. With our assumption, the agent assumed that there still existed a path. To follow this path, it kept rotating to the right until an real valid path was found. By updating and following the path in each step, the agent would eventually reach the target.

## Description of Implementation

The link to our source code is < <https://github.com/Yang-Yuan/AnimalAI-Olympics/tree/master/agents> >. There are also some other source code in this repository, but all the files that were used in the docker images tested online are in the <agents> directory. The A1 agent is defined in the file <reflexAgent.py>, and the A2 agent is defined in <handcraftedAgent.py>. Since A1 agent is fairly simple, we will only describe A2 agent in this section.

A2 agent includes four functional modules: perception, strategy, action\_state\_machine and chaser. After the agent is called, it first uses the perception module to process the raw visual input to label each pixel as food, obstacles, danger or sky. Next, the agent uses the strategy module to generate the action. To reduce the complexity, the strategy module is built on a finite state machine, called action\_state\_machine in our code. The state transitions are determined by the labeled visual input from the perception module. The action for the agent to take is set in all the callback functions of state transitions. The chaser, generating the actions to chase a target, was supposed to be part of these callback functions. However, because of its complexity, we implemented in a separate module.

## Conclusion

By comparing two agents, we showed that memory module, object-permanence and assumption ability are critical to the performance of our animal AI agents. In our non-learning implementation, we hard-coded the prior knowledge to generate memory, object-permanent representation and an optimistic assumption for path planning. They all inspired us to think further about how these faculties and prior knowledge can be learned by agents in the future.

1. These two agents are Submission 42616 and Submission 45109 in the EvalAI website. The former is A1 and the latter is A2. [↑](#footnote-ref-1)