Environment "Single Agent Catches the Pigs"

Documentation

Author: Shuo Jiang (jiang.shuo@husky.neu.edu)

Introduction

The document introduces an environment for a single agent study as one agent is trying to catch a moving (or fixed) pig in the pigsty. The code is implemented in python and provided with simple interfaces. The environment is decoupled with learning method so reader can try any algorithms on.

Scene Description

The task of the environment is explained as below

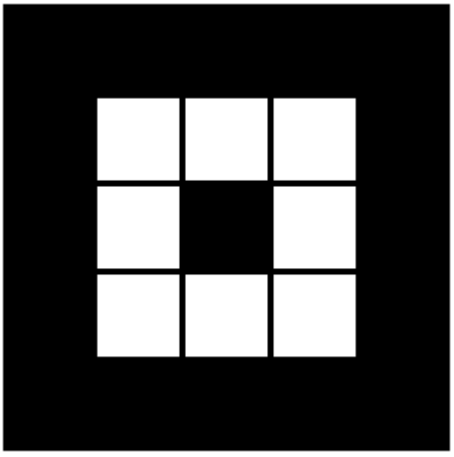

Figure 1

The environment works like a pigsty, one agents will run in the pigsty and try to catch the moving (or fixed) pig. The agent 1 will be marked using color red, and will be shown as red symbol in the maze, and the pig will be denoted using color green like:

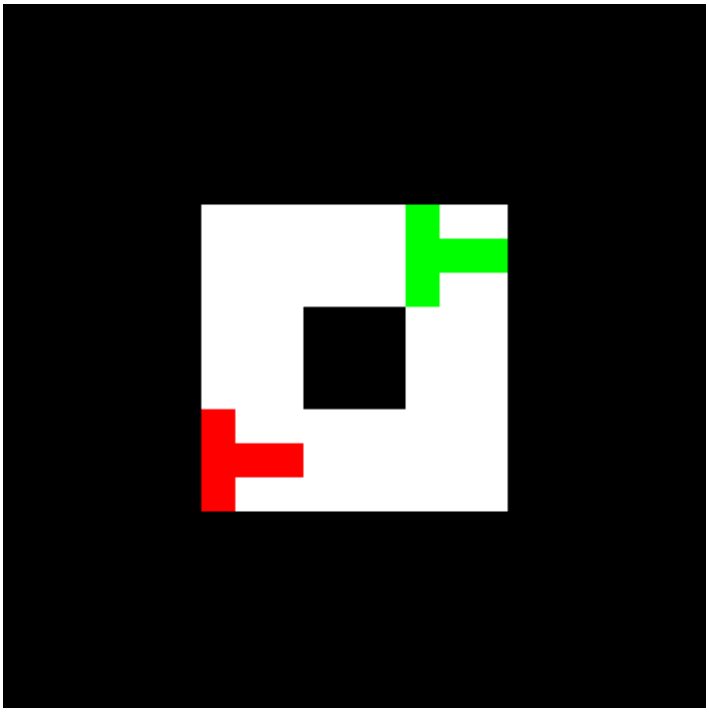

Figure 2

blocks in black are obstacles that agent could not step on and white ones are free spaces. The coordinate system is shown as

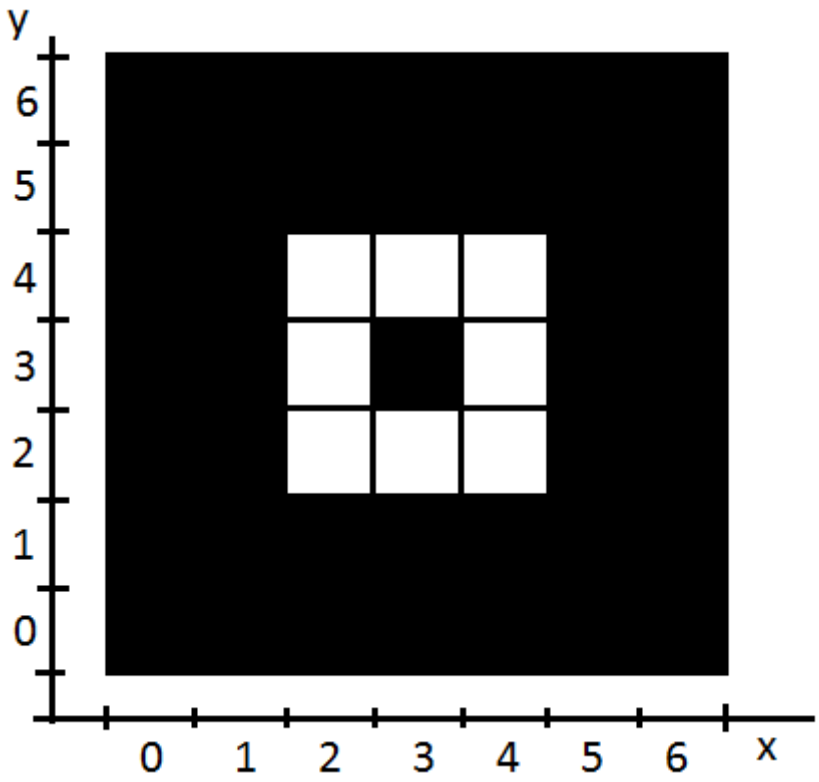

Figure 3

The agent and the pig will be at random positions at the beginning with no overlapping. When the agent catches the pig, the environment will reset and the two entities will be set at random positions again.

There are 5 possible actions for the agent

Figure 4

when the agent moves, it will move to the free space of 1 distance near it. However, when there is a block or the other agent is in its way, the action won't work.

Actions for the pig are similar but without catch action

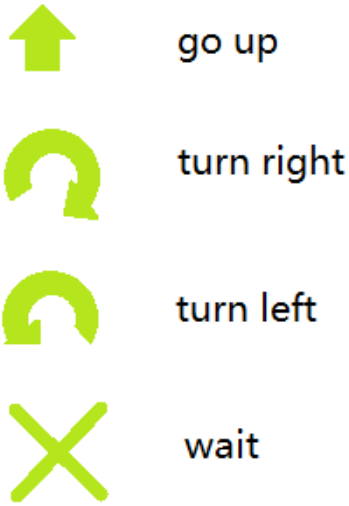

Figure 5

When the agents uses action catch, the pig must be in the 1 nearby position of any one. (means the only when agents are close enough that catch action can work, also the pig should be at the front of agent orientation. When the pig is caught, the environment resets and pig and agent will be randomly relocated. And the agent gets a positive reward of 200 and pig gets negative reward of -200. Other reward like no matter when agent or pig choose an action besides wait, it will be rewarded -1 as some sort of energy consumption. And move action bumping to the wall will cause -20 reward and using catch but failed to catch the pig will be rewarded -50

Also the environment can be locally observed by each thing. Each entity (agent and pig) has 4 orientations, and they can only observe about 90 degree of environment in front of it. Orientation will be indicated by an integer 0, 1, 2, 3 as shown in graph.

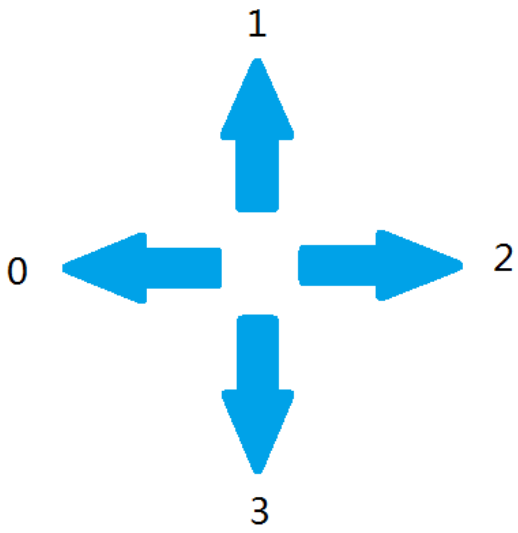

Figure 6

In the plot, the orientation will be shown using a "T" shape for different orientation. Agent will use red "T" and pig uses green "T".

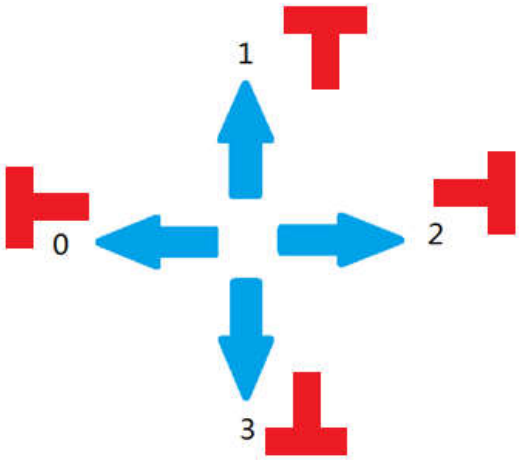

Figure 7

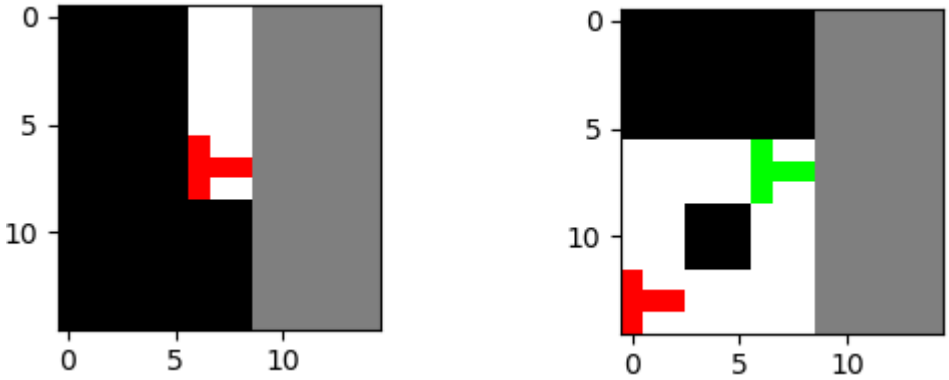

Figure 8

An example of observation is shown in Figure 8. From left to right are observations of agent 1 and pig. As shown in the graph, agent 1 is facing left and pig is facing left. The light grey areas are unobserved areas.

So the observation is an image, with form (width = 15, height = 15, rgb_channel = 3)

Class Organization

The environment is programmed in python 3.6 and has very simple interfaces. The file contains the environment is "env_SingleCatchPigs.py" and a class " EnvSingleCatchPigs
" is defined.

The class has the following interfaces:

```
__init__(self, if_PO)
reset(self)
get_agt1_obs(self)
get_pig_obs(self)
get_global_obs(self)
step(self, action1, action_pig)
plot_scene(self)
set_agent_at(self, tgt_pos, tgt_ori)
set_pig_at(self, tgt_pos, tgt_ori)
```

`__init__(self, if_PO)`

initialize the object, important variables are

`agt1_pos`

`agt1_ori`

`pig_pos`

`pig_ori`

These can be set as a list as agt_pos=[3, 1], agt_ori=0. The variable `if_PO` is a bool variable, means Partially observable. If is False the observation of each entity will be the global map and if it is set True, the entity can only observe locally. as

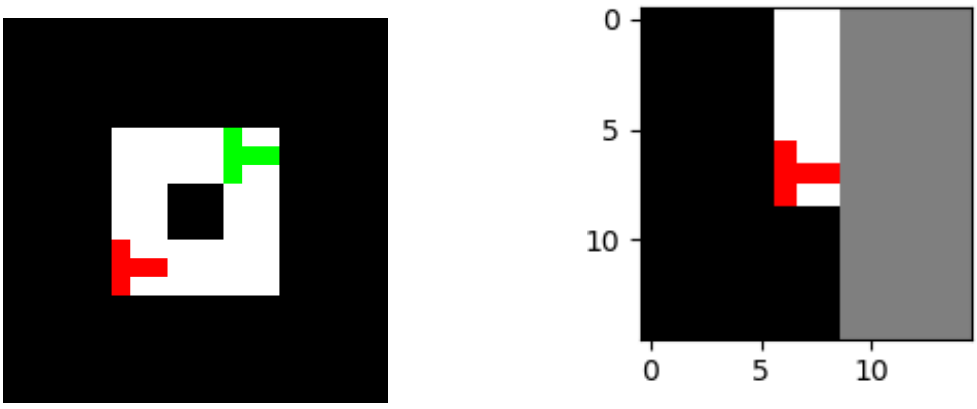

Figure 9

Example as fully observable (left) and partially observable (right)

get_agt1_obs(self):

These function return the current vision of agent 1 by returning a image. Depending on if the environment is set as partially observable. The return could be left in Figure 9 or right in Figure 9.

`get_global_obs(self)`

This function always returns the left of Figure 9.

`step(self, action1, action_pig)`

The function update the environment by feeding the actions for agent1 and agent2 and pig. The actions are expressed by integers as shown in Figure 4 and 5.

reset(self):

This function randomly relocates the two agent and the pig

plot_scene(self):

This function plot the current state of the whole environment, and uses three subplots to show the views of each agent and pig

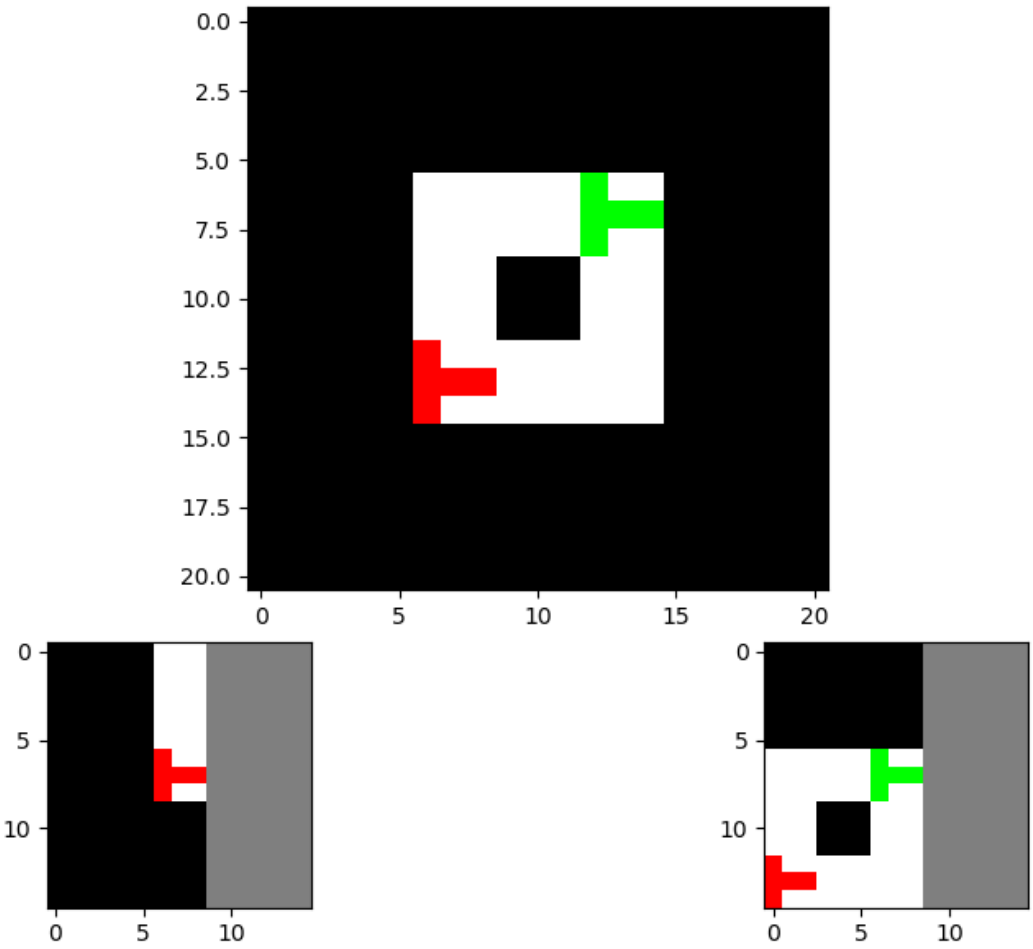

Figure 10

The global positions of agent1 (red) and pig (green) are shown as figure 10 top. The visions of things are listed at bottom, from left to right are view of agent 1 and view of pig. Unobserved area are shown using light grey.

`set_agent_at(self, tgt_pos, tgt_ori)`
This function is used to set the position of agent at target position with target orientation. For example, If you want to set the agent at [2, 2] and to left. Call the function as
`set_agent_at(self, [2, 2], 0)`
However if the target position is not free space, this function will not change anything.

Example

Here is an example using test function, and it is in "test_FindGoals.py"

```
from env_SingleCatchPigs import EnvSingleCatchPigs
import random

env = EnvSingleCatchPigs(True)
max_iter = 10000
env.set_agent_at([2, 2], 0)
env.set_pig_at([4, 4], 0)
for i in range(max_iter):
    a_1 = random.randint(0, 4)
    a_pig = random.randint(0, 3)
    print("iter= ", i, env.agt1_pos, env.pig_pos, env.agt1_ori, env.pig_ori,
a_1)
    env.plot_scene()
    reward_1, reward_pig, obs_1, obs_pig = env.step(a_1, a_pig)
    if reward_1 > 0:
        print("agent 1 finds goal")
        env.plot_scene()
```