Environment "Catch the Pigs" Documentation

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

Introduction

The document introduces an environment for multi agent study as two agents are trying to catch a moving 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.

Scenario Description

The task of the environment is explained as below

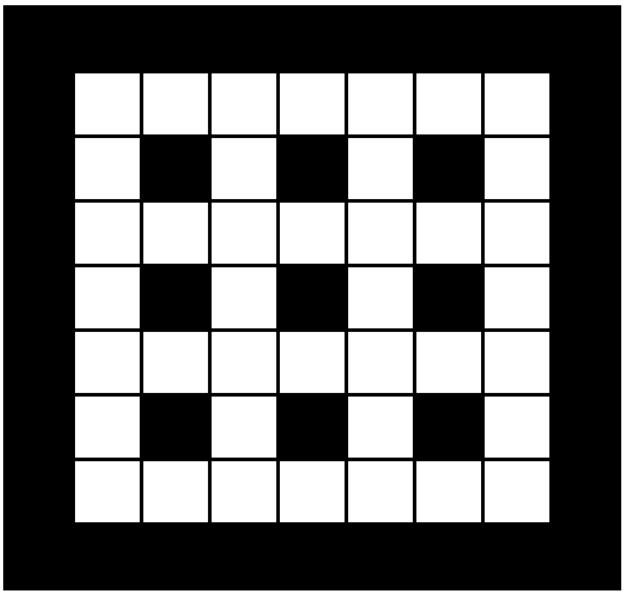


Figure 1

The environment works like a pigsty, two agents(agent 1 and agent2) will run in the pigsty and try to catch the moving pig. The two agents, agent 1 and agent2 will be denoted using color red and blue, and will be shown as red and blue rectangles in the maze, and the pig will be denoted using color green like:

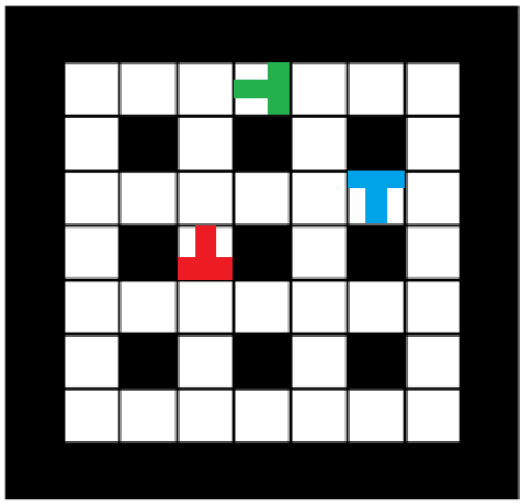


Figure 2

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

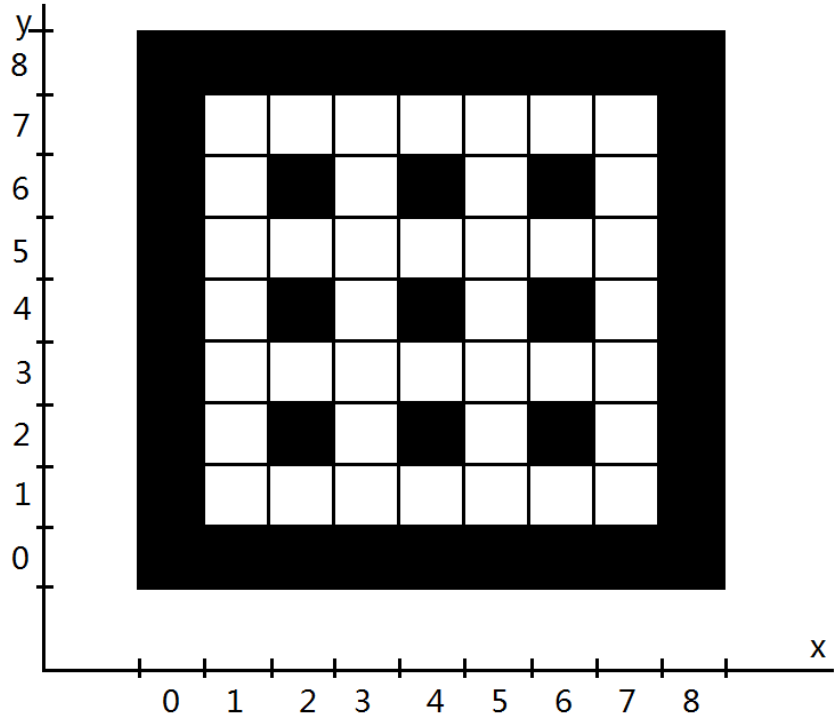


Figure 3

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

There are 5 possible actions for the two agents

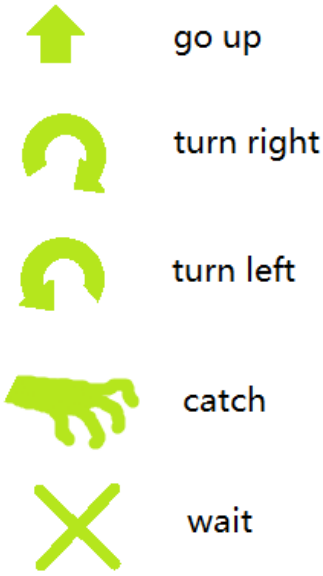


Figure 4

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

The pig will move randomly in the pigsty.

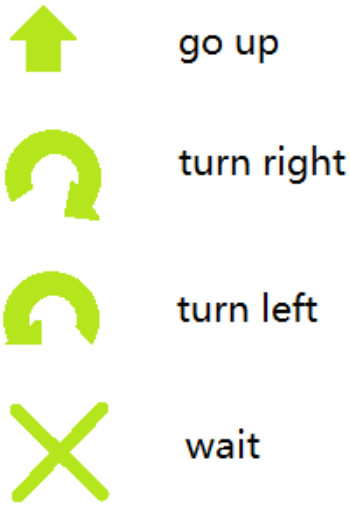


Figure 5

We assume the pig is strong and only when two agents simultaneously use "catch" action, will the pig be caught. When the two agents use action catch, the pig must be in the 1 nearby position of any one and the agents should be facing towards the pig. (means the only when agents are close enough that catch action can work, and simultaneous catching can succeed).

When the pig is caught, the environment resets and pig and agents will be randomly relocated. And the each agent gets a positive reward of 500. 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 agent, each agent 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.

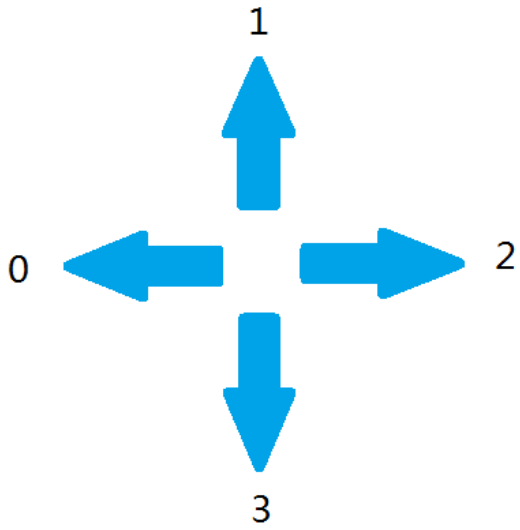


Figure 6

Also each of the entities (agents and pig) will shown using a "T" symbol representing its orientation in the plot. As

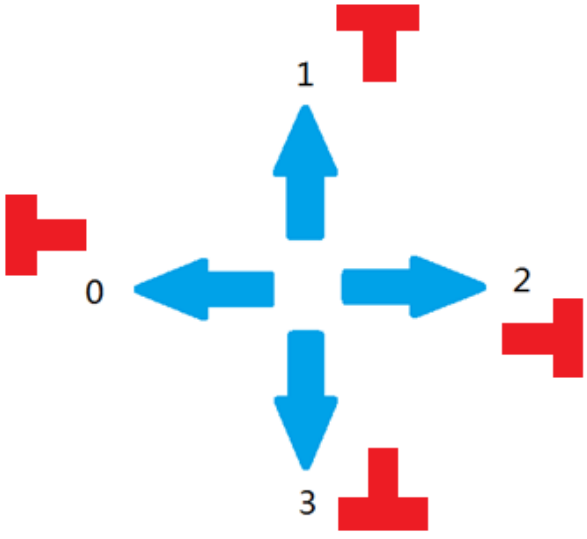


Figure 7

Reader can imagine the pig is presented as green "T" and agent 2 is presented as blue "T"

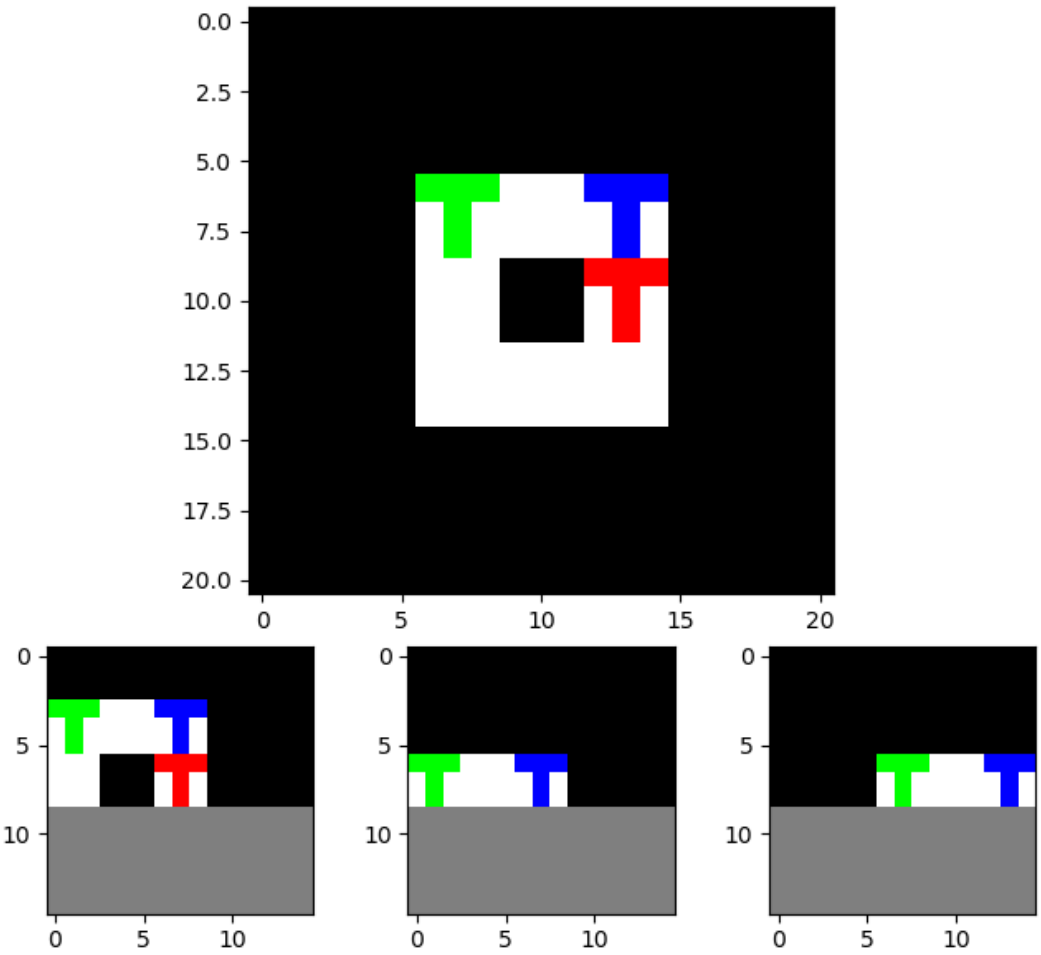


Figure8

An example of observation is shown in Figure 8. On the top is the global view with full observation, but usually we do not use it for the POMDP problem. On the bottom from left to right are observations of agent 1, agent 2 and pig. As shown in the graph, agent 1 is facing up, agent 2 is facing up and pig is facing up. The light grey areas are unobserved areas, meaning the entity is only able to see the environment in front of it.

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

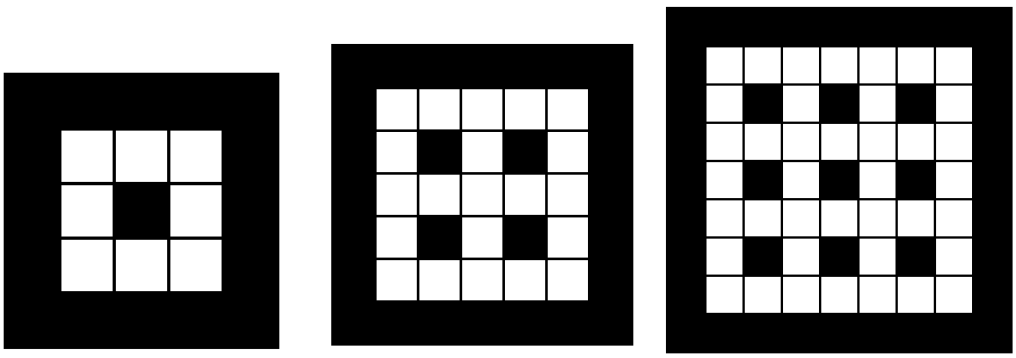
Class Organization

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

The class has the following interfaces:

```
__init__(self, size, if_PO)
reset(self)
obs_list = get_obs(self)
obs = get_global_obs(self)
reward_list, done = step(self, action_list)
plot_scene(self)
set_agt1_at(self, tgt_pos, tgt_ori)
set_agt2_at(self, tgt_pos, tgt_ori)
set_pig_at(self, tgt_pos, tgt_ori)
render()
```

`__init__(self, size, if_PO)`
initialize the object, important variables are
`agt1_pos`
`agt1_ori`
`agt2_pos`
`agt2_ori`
`pig_pos`
`pig_ori`
These are programmed by a list as agt_pos=[3, 1], agt_ori=0
Also, the parameter size controls how large the map is, it should be assigned as an odd integer no smaller than 5. and the generated maps are as



size=7 size=9 size =11

get _obs(self):

These function return a list of current visions of each agent as an image, with form (width = 5, height = 5, rgb_channel = 3) as shown in Figure 8 bottom. obs_list = [obs1, obs2]

step(self, action_list):

The function update the environment by feeding the actions for agent1 and agent2. The actions are expressed by integers as shown in Figure 4 and 5. action_list = [action1, action2]. The return is reward_list = [reward1, reward2] and done

reset(self):

This function randomly relocates the two agents 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

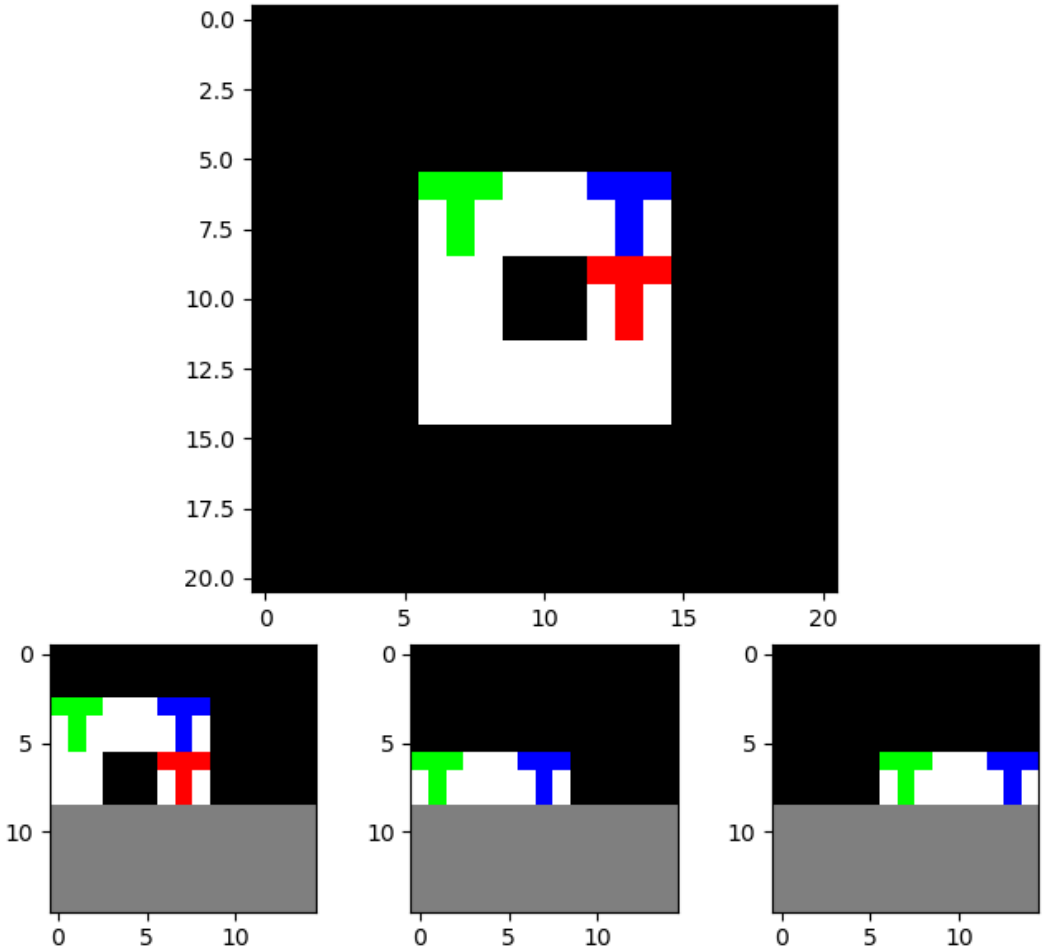


Figure 9

The global positions of agent1 (red) and 2 (blue) and pig (green) are shown as figure 8 top. The visions of things are listed at bottom, from left to right are view of agent 1, view of agent 2 and

view of pig. Unobserved area are shown using light grey.

`set_agt1_at(self, tgt_pos, tgt_ori)`
This function is used to set the position of agent 1 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_agt1_at(self, [2, 2], 0)`
However if the target position is not free space, this function will not change anything.

`render(self)`
This function plot scene in animation, call in each step

Example

Here is an example using test function, and it is in "test_CatchPigs.py". The actions for agents and pig are random chosen, click and run.

```
from env_CatchPigs import EnvCatchPigs
import random

if __name__ == '__main__':
    env = EnvCatchPigs(7, True)
    max_iter = 10000
    for i in range(max_iter):
        action1 = random.randint(0, 4)
        action2 = random.randint(0, 4)
        action_list = [action1, action2]
        obs_list = env.get_obs()
        obs1 = obs_list[0]
        obs2 = obs_list[1]
        print("iter= ", i, env.agt1_pos, env.agt2_pos, env.pig_pos,
env.agt1_ori, env.agt2_ori, 'action', action1, action2)
        env.render()
        reward_list, done = env.step(action_list)
        #env.plot_scene()
        if reward_list[0] > 0:
            print("iter= ", i)
            print("agent 1 finds goal")
```