Environment "Drones Monitoring" Documentation

Author: Shuo Jiang (jiang.shuo@husky.neu.edu)
License: Northeastern University, Lab for Learning and Planning in Robotics(LLPR)

Introduction

The document introduces an environment for a group of UAVs monitoring a certain area. 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

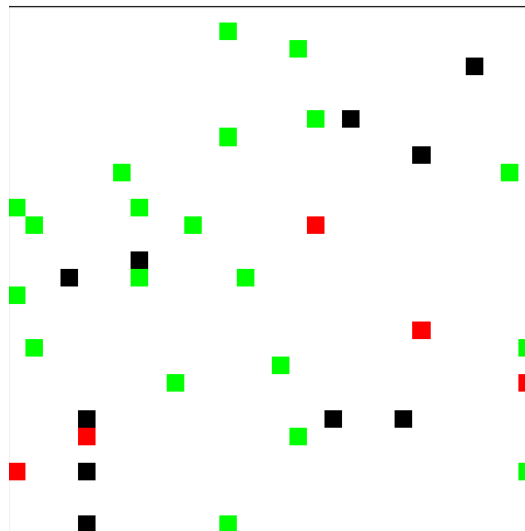The task of the environment is explained as below

Figure 1

The environment works like a squared area, some humans are randomly walking in the area as shown in figure 1. Humans are denoted as red rectangles, and there are trees (green) and walls (black) in the environment (potentially used for localization problem). Humans can walk in the free space as white blocks in the area but should not bump into the walls and trees.

The above description is about what happens on the ground. Also, there is a group of drones flying on the sky using camera monitoring what happens on the ground. The drones (UAVs) can fly freely in the given area (overlapping the same position is possible). Each drone has a local vision defined by a radius, and things it can observe are only in a local area, or we say partially observable. The typical vision of an agent is shown as
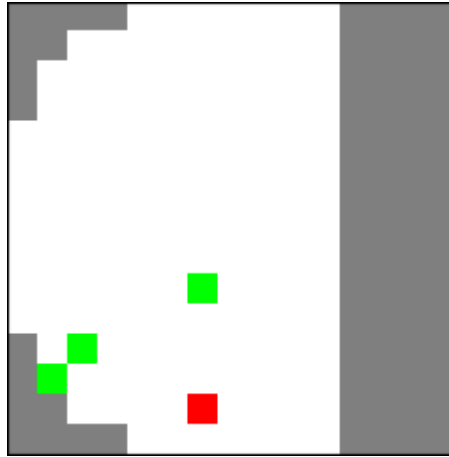
Figure 2

The drone cannot see itself or other drones in its vision but only the things on the ground. Unobservable areas which is beyond max vision radius are shown in light grey and areas out of the square space will also be shown as light grey blocks. So in Figure 2, a typical vision of one drone is provided, usually is an circular area, however this drone is at the right boarder so the right part of the vision is in light gray. reader can see there are three trees (green) and one human (red) observed.

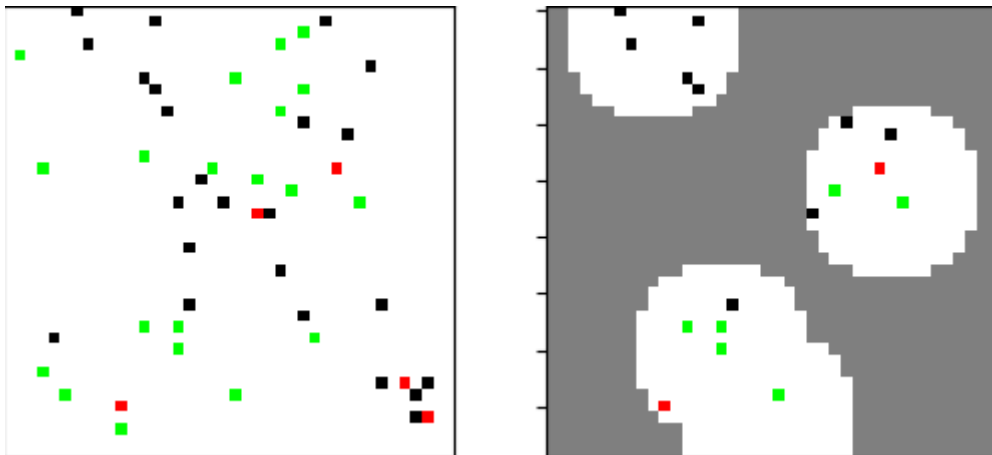We can also see the joint observation for all drones as

Figure 3

The left part of figure 3 is the fully observed ground status and the right part is the joint vision of 4 drones. So reader can see the joint vision is only a subset of what happens in the left plot.
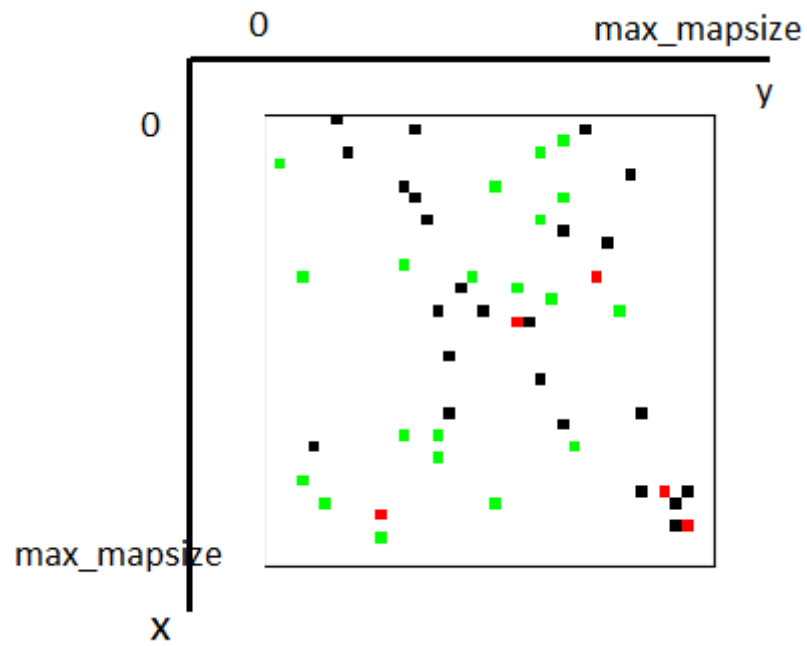
The coordinate system of the environment is as

Figure 4

There are 5 possible action for the drones

	go up	0
go down	1
go left	2
go right	3
wait	4

Figure 5

When the drone moves, it will move to the place of 1 distance near it, so as the humans on the ground. The action is denoted by a integer from 0 - 4 as shown in right part of figure 5.

This environment provides with no reward. So users can define their own reward function. Here I

can give some insight about how the reward function should be.

1. The goal for the swarm of drones is monitoring the humans on the ground, which means the humans should appear in the joint vision of drones.

2. When the number of human is larger than the number of drones, which means it sometimes can be impossible to monitor all the humans at the same time. Under this circumstance, monitoring more humans will get higher reward. So the drones should reduce the overlapping areas of monitoring.

3. Due to the partial observability, the number of humans will be unknown to drones, so trying to explore the whole area to ascertain the number of humans is important.

Class Organization

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

There are two classes also defined as "Human" and "Drones".

For class "Human", there is a member variable

`self.pos = pos`

It is a list of 2 elements and can be set with a position as:

pos = [2, 4]

For class "Drones", there is a member variable

`self.pos = pos`

It is a list of 2 elements and can be set with a position as:

pos = [2, 4]

also, there is a variable "view_range", which is the radius of local observation.

In the top class " EnvDrones ", there are several important variables as:

```
map_size (integer) the max_size of map
drone_num (integer) the current number of drones, which can vary as new drones
        joins the group
tree_num (integer) the current number of trees
human_num (integer) the current number of humans, which can vary as new human
        joins the group
human_list (list) contains all the current human entities
drone_list (list) contains all the current drone entities
start_pos (list of two elements) is the start position, we may assume that
        at the beginning, all drones are located at the right bottom corner
        of the map.
```

Some of the important member functions are:

```
__init__(self, map_size, drone_num, view_range, tree_num, human_num)
```

In this function, user should provide with 5 integers as parameters of generating the environment.

`get_full_obs(`self`)`
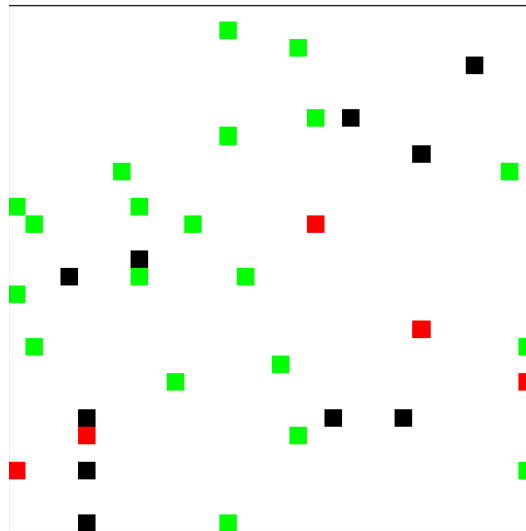The function returns a rgb image of size (`map_size, map_size, 3`) as

Figure 6

`get_drone_obs(`self, drone`)`
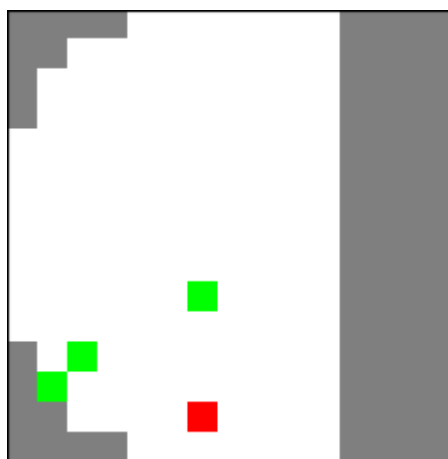This function returns a rgb image of size (`view_range, view_range, 3`) as

Figure 7

The parameter "`drone `" is an entity of class "Drones", and usually is a component in drone_list of "env". So user can call this function as
env.get_drone_obs(env.drones_list[0])
This will return the vision of the first drone in the list.

`get_joint_obs(`self`)`
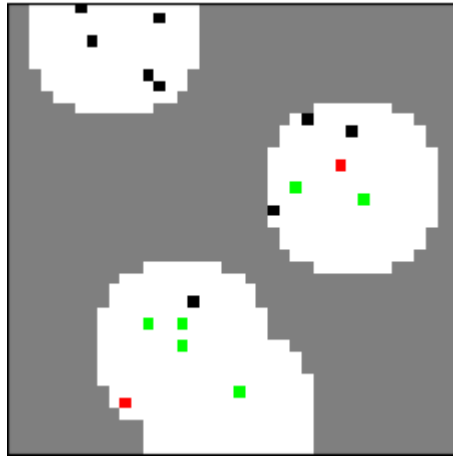This function returns a rgb image of size (`map_size, map_size, 3`) as

Figure 8

`rand_reset_drone_pos(self)`

This function will relocate all the drones at random positions in the area.

`step(self, human_act_list, drone_act_list)`

This function will update the positions of humans and drones by giving their corresponding actions as "human_act_list" and "drone_act_list". "human_act_list" and "drone_act_list" are two list variable, and the number of elements should be the same as "human_num" and "drone_num" variables in "env". The elements are integers from 0-4 as in

Figure 9

So if reader wants humans and drones walking randomly in the space should do

```python
for i in range(env.human_num):
        human_act_list.append(random.randint(0, 4))
```

```python
drone_act_list = []
for i in range(env.drone_num):
    drone_act_list.append(random.randint(0, 4))
env.step(human_act_list, drone_act_list)
```

Example

Here is an example using test function, and it is in "test_Drones.py", just click "run" and you shall see how drones are monitoring under a random policy.

```python
from env_Drones import EnvDrones
import random
import matplotlib.pyplot as plt
from matplotlib.gridspec import GridSpec


env = EnvDrones(50, 4, 10, 30, 5)   # map_size, drone_num, view_range,
tree_num, human_num
env.rand_reset_drone_pos()

max_MC_iter = 100
fig = plt.figure()
gs = GridSpec(1, 2, figure=fig)
ax1 = fig.add_subplot(gs[0:1, 0:1])
ax2 = fig.add_subplot(gs[0:1, 1:2])
for MC_iter in range(max_MC_iter):
    print(MC_iter)
    ax1.imshow(env.get_full_obs())
    ax2.imshow(env.get_joint_obs())

    human_act_list = []
    for i in range(5):
        human_act_list.append(random.randint(0, 4))

    drone_act_list = []
    for i in range(4):
        drone_act_list.append(random.randint(0, 4))
    env.step(human_act_list, drone_act_list)
    plt.pause(.5)
    plt.draw()
```