Cleaning Agent

Group 3

By: Attah Khan, Eshan Gaur,
Neeraj Varshney and Tithi Gupta

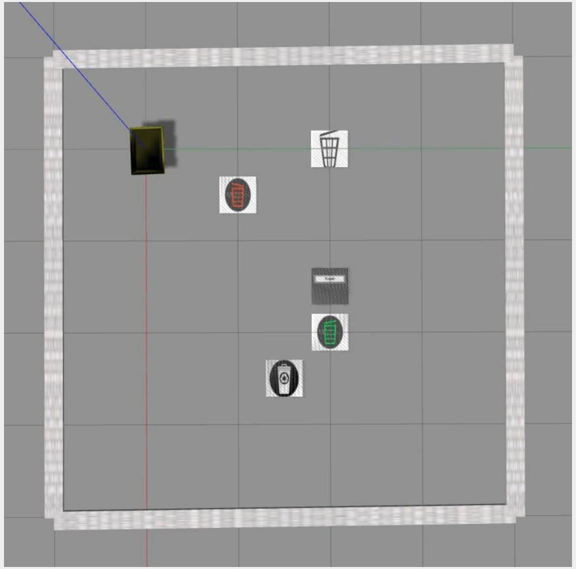
Presentation Outline

- Project Description
- End-2-End demonstration
- Detailed explanation of the approach

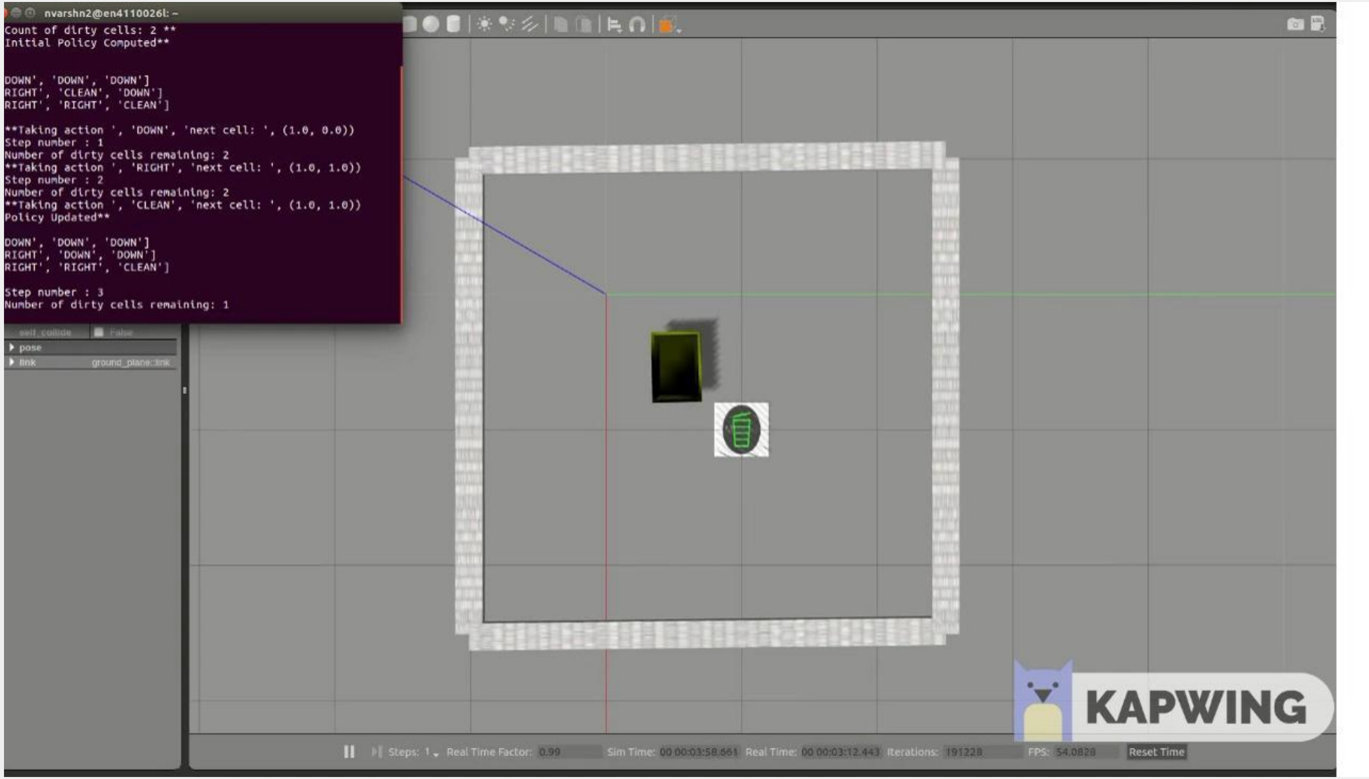
Project Description

- Objects -
  - Agent
  - Dirt spread out randomly with in the grid
- Actions -
  - Move UP
  - Move DOWN
  - Move LEFT
  - Move RIGHT
  - CLEAN

  *Note - Actions are stochastic
- Objective - To clean all the dirty cells in an optimal way



Demo



```
Count of dirty cells: 2 **
Initial Policy Computed**


DOWN', 'DOWN', 'DOWN']
RIGHT', 'CLEAN', 'DOWN']
RIGHT', 'RIGHT', 'CLEAN']

**Taking action ', 'DOWN', 'next cell: ', (1.0, 0.0))
Step number : 1
Number of dirty cells remaining: 2
**Taking action ', 'RIGHT', 'next cell: ', (1.0, 1.0))
Step number : 2
Number of dirty cells remaining: 2
**Taking action ', 'CLEAN', 'next cell: ', (1.0, 1.0))
Policy Updated**

DOWN', 'DOWN', 'DOWN']
RIGHT', 'DOWN', 'DOWN']
RIGHT', 'RIGHT', 'CLEAN']

Step number : 3
Number of dirty cells remaining: 1
```

self_collide □ False

pose

link ground_plane::link

Steps: 1 Real Time Factor: 0.99 Sim Time: 00 00:03:58.661 Real Time: 00 00:03:12.443 Iterations: 191228 FPS: 54.0828

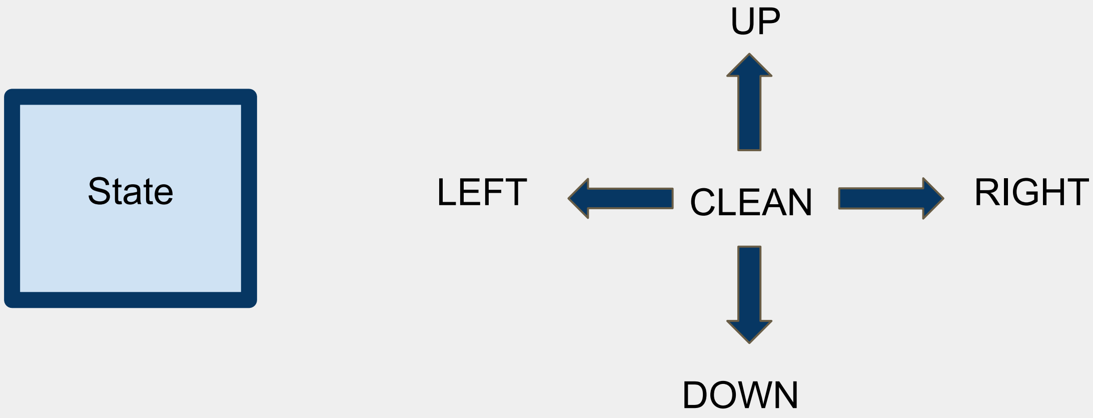
Project Description cont.

The objective of this project is to learn a policy for an agent that operates in a grid-world environment where it has to clean the cells that contains dirt. Since the environment is modeled using MDPs, the actions are stochastic in nature and agent has associative probability of successfully executing every action. Dirt is spread out randomly in the environment.

To summarize, we have a ***static, fully observable, stochastic*** environment.

Approach

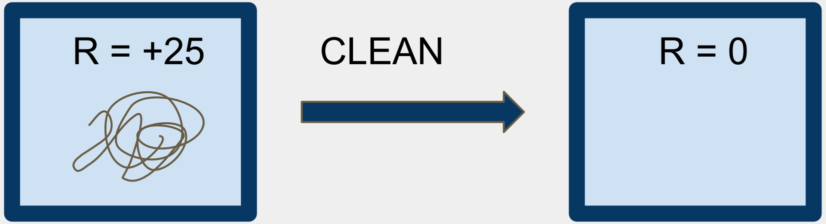
- Model this problem as an MDP.
  - Each cell is represented by a state
  - 5 possible actions in each state
  - Actions are stochastic with associated probability
- Use Value / Policy Iteration to find optimal action for every state.



Approach cont.

What happens after cleaning a dirty cell?

Now, the reward for clean action in that particular cell no more exists.
Use policy iteration to refine the policy.



Future work

- Handling dynamically appearing dirt:

  Before executing an action check if the environment has changed since the last time. If yes, compute the policy again.

- Adding obstacles (for e.g. blocking a few movements in certain states)

Learning Outcome

- Usage of MDP to solve the problem statement.
- Manipulation of the Gazebo environment for a new environment.
- Learned to work with ROS.

Supplementary Slide 1

- Why use Policy iteration and not Value iteration again to find the policy after cleaning a dirty cell?
  - It is intuitive that the optimal action for the cells far off from the recently cleaned cell won’t change. Optimal action for only the neighboring cells is expected to change. Hence, using Policy iteration would reach convergence faster.
- How Actions are modeled?
  - Mapping for each orientation and action. For e.g. If robot is facing WEST and action is to move DOWN then the sequence of low level actions would be ["TurnCW", "TurnCW", "moveF"]
- Since actions are stochastic, Can the bot move out of the grid?
  - No, In a state where taking an action bumps the agent to a nearby wall, doesn’t change the state of the agent, i.e., the agent ends up in the same cell.

Supplementary Slide 2

- How the objects file look like?

  ```
  {
  "grid_size": 6,
  "dirts": {
          "dirt_4": {"loc": [1.0, 2.0]},
          "dirt_5": {"loc": [3.0, 4.0]},
          "dirt_1": {"loc": [5.0, 3.0]},
          "dirt_2": {"loc": [4.0, 4.0]},
          "dirt_3": {"loc": [0.0, 4.0]}
  }}
  ```