CS5335 Final Project

By: Amit Mulay

Charles Denhart

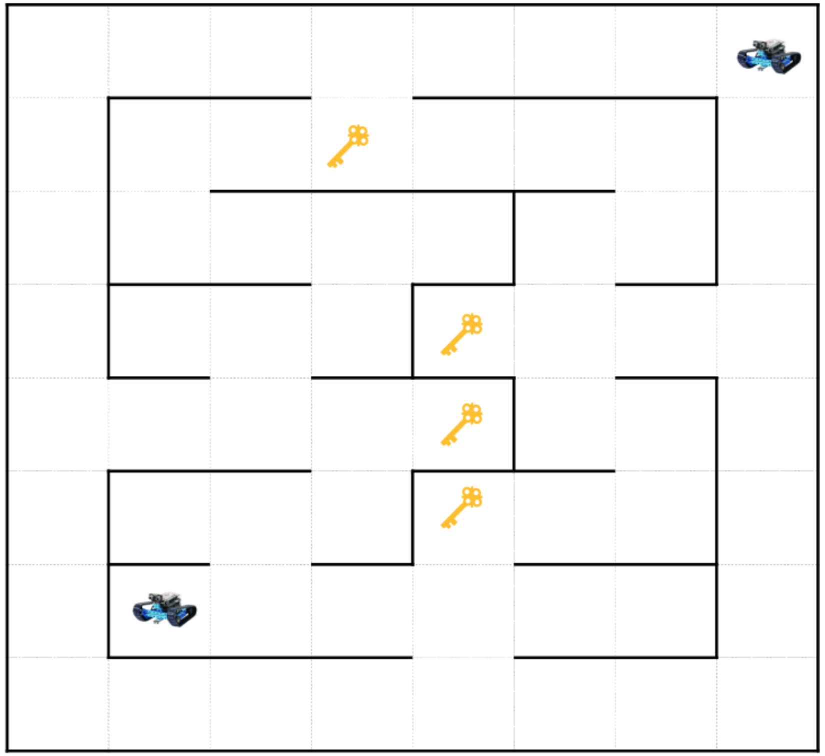
Kelly King

Pablo Gomez

Original Goals: Maze Prison Escape

Implement a Maze Prison Escape Game.

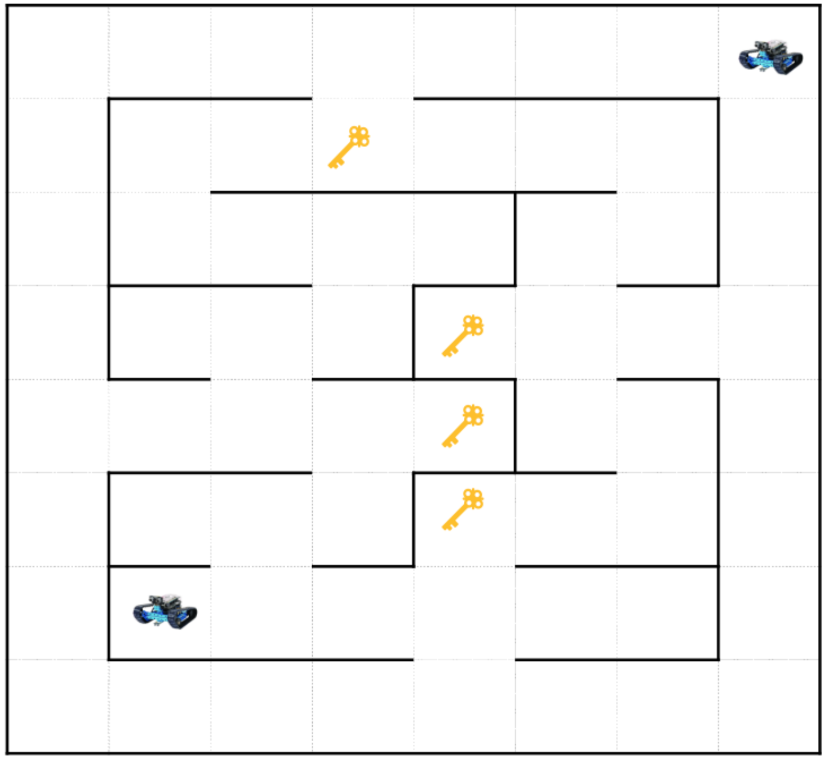
Similar concept to PAC-MAN: main agent attempting to achieve some goals and a secondary agent acting as an adversary.



- Prisoner Goals/Behavior:
  - Escape Maze
  - Collect all the keys
  - React to lights and sound to avoid the patrol robot
- Patrol Goals/Behavior:
  - Patrol perimeter of the maze
  - Drive into the maze when it found an opening
- Maze configuration:
  - Beginning and End
  - At least 2 paths to each key

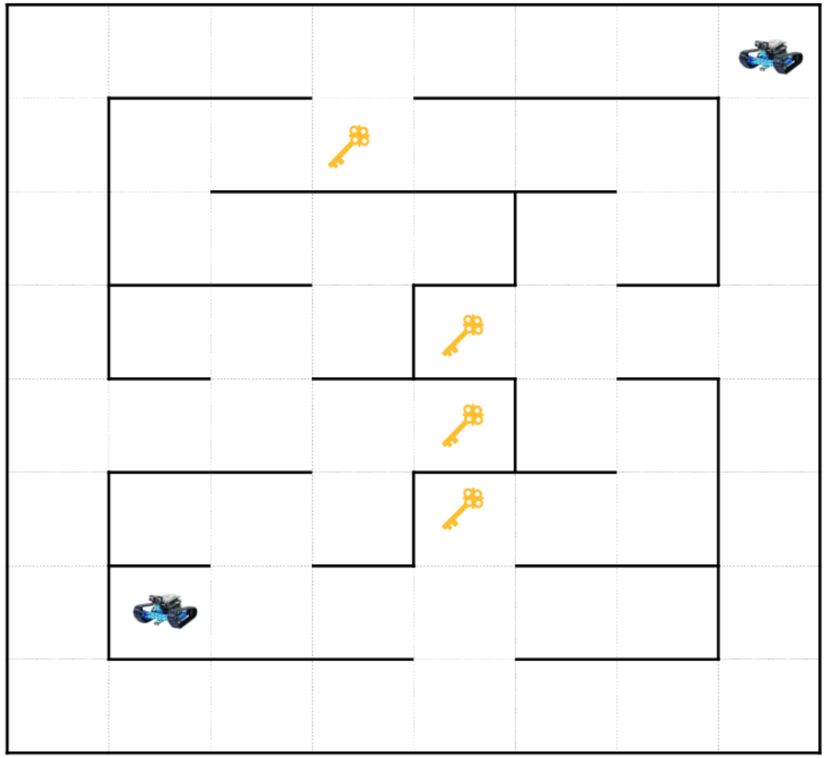
Considerations

- How to indicate the prisoner has escaped?
  - Map of the maze: robot stops when in escape position
- How will the Prisoner move around the maze?
  - Wall following would be the simplest solution to implement, but not conducive to avoiding the patrol's path.
- Method for collecting the keys?
  - Using the line follower sensor



Considerations cont.

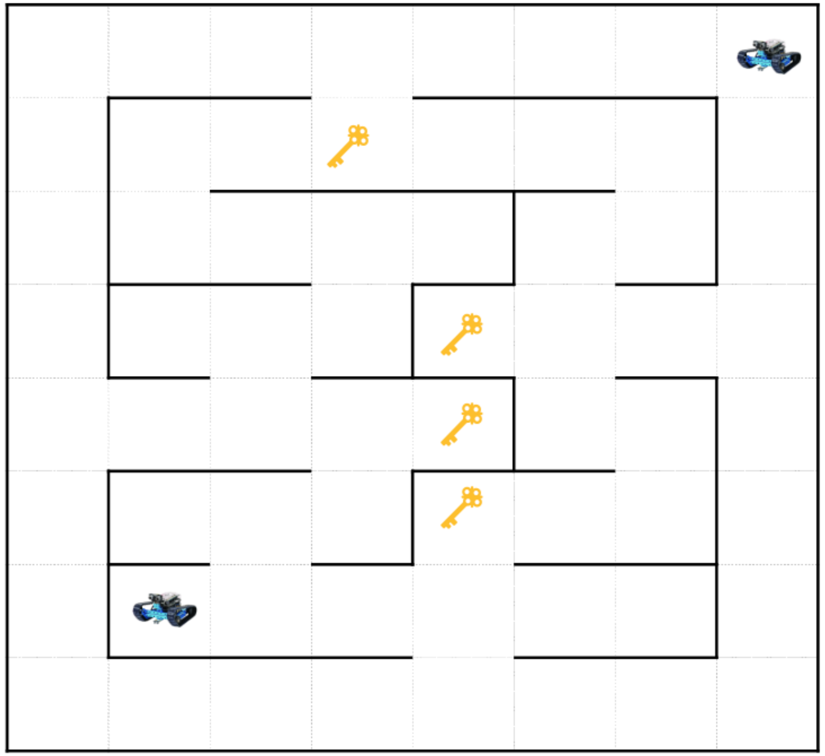
- How does the prisoner react to the patrol?
  - Patrol uses led-ring and buzzer to produce light and sound that the prisoner can detect
- How will the Patrol move?
  - Using the line follower sensor
- How will the patrol know to enter the maze?
  - Option A: Map the maze – turn and reverse at certain locations
  - Option B: Add an extra sensor – detect openings on the inside wall



Original Goals: Maze Prison Escape

Implement a Maze Prison Escape Game.

Similar concept to PAC-MAN: main agent attempting to achieve some goals and a secondary agent acting as an adversary.

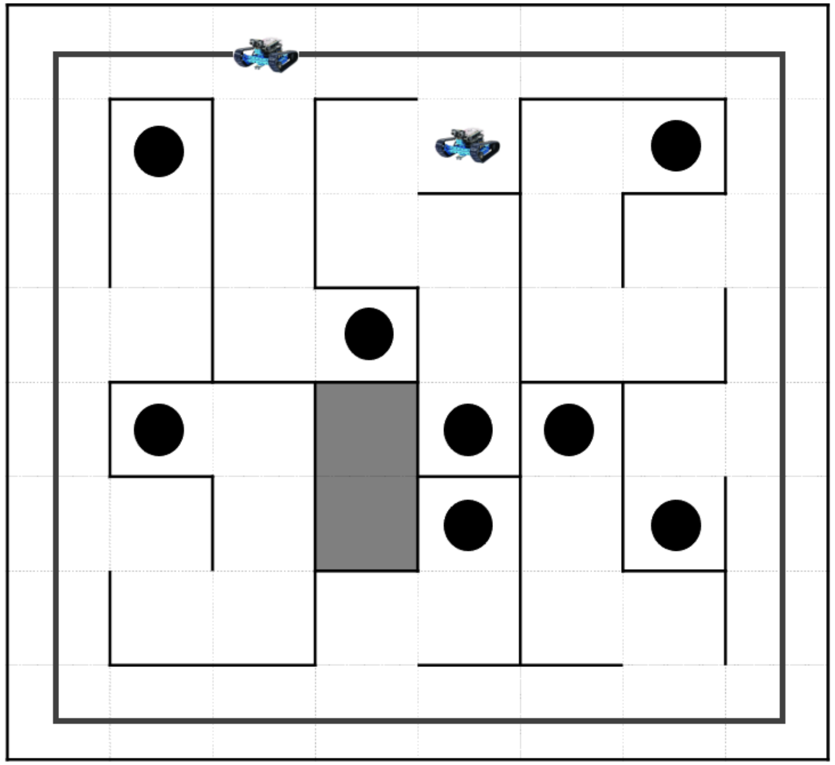


- Prisoner Goals/Behavior:
  - Escape Maze
  - Collect all the keys
  - React to lights and sound to avoid the patrol robot
- Patrol Goals/Behavior:
  - Patrol perimeter of the maze
  - Drive into the maze when it found an opening
- Maze configuration:
  - Beginning and End
  - At least 2 paths to each key

Revised Goals: Maze Prison Escape

Implement a Maze Prison Escape Game.

Similar concept to PAC-MAN: main agent attempting to achieve some goals and a secondary agent acting as an adversary.



- Prisoner Goals and Behavior:
  - ~~Escape Maze~~
  - Collect all the keys
  - ~~React to lights and sound to avoid the patrol robot~~
- Patrol Goals:
  - Patrol perimeter of the maze
  - ~~Drive into the maze when it found an opening~~
  - Sleep and chase for randomized time intervals (10-20sec)
- Maze configuration:
  - ~~Beginning and End~~
  - ~~Keys are reachable without encountering Patrol~~
  - Keys are in "rooms" that require crossing the Patrols Path

Implementation

Prisoner

- □ Wall Following
- □ Reinforced Learning

Patrol

- □ Line Following
- □ Randomized Behavior

Academic Paper Technique:
Reinforcement Learning (RL)

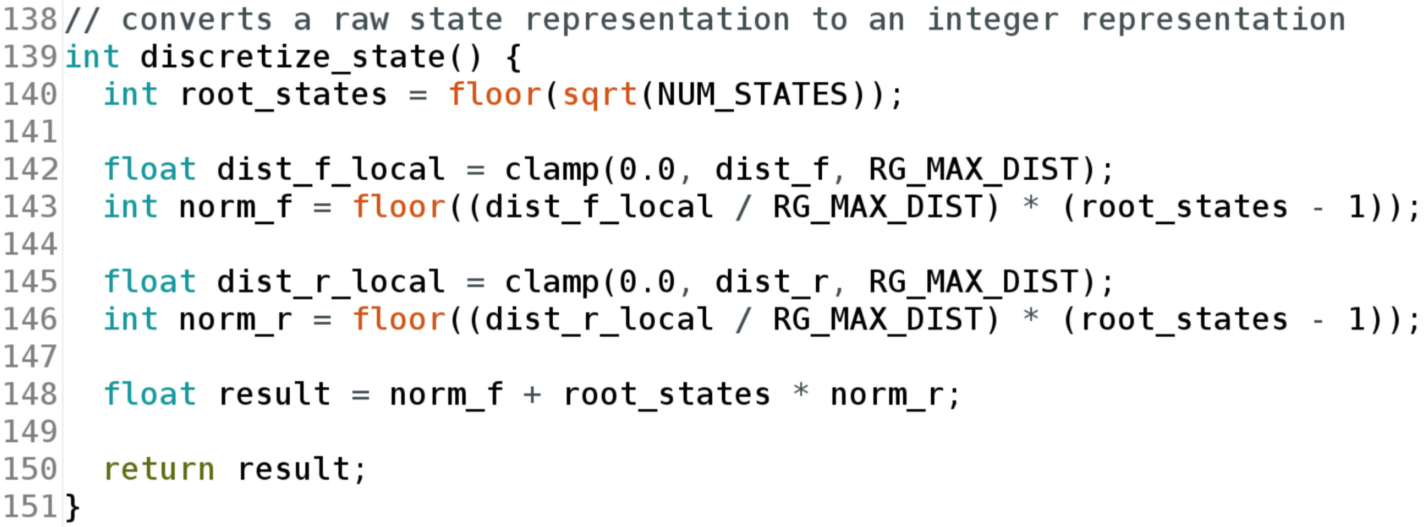
- Q-Learning: $Q^*(s,a) = E\left[R(s,a) + \gamma \max_{a'} Q^*(s',a')\right]$
- Sparse reward function (vs. dense reward function)
- Uses original function approximation algorithm (HEDGER)
- Two tasks:
  1. Corridor following
  2. Obstacle avoidance with a target

Smart, W. D., &amp; Kaelbling, L. P. (n.d.). Effective Reinforcement Learning for Mobile Robots. Retrieved from http://www.sci.brooklyn.cuny.edu/~sklar/teaching/f06/air/papers/smart-icra2002.pdf

Deviations from Paper

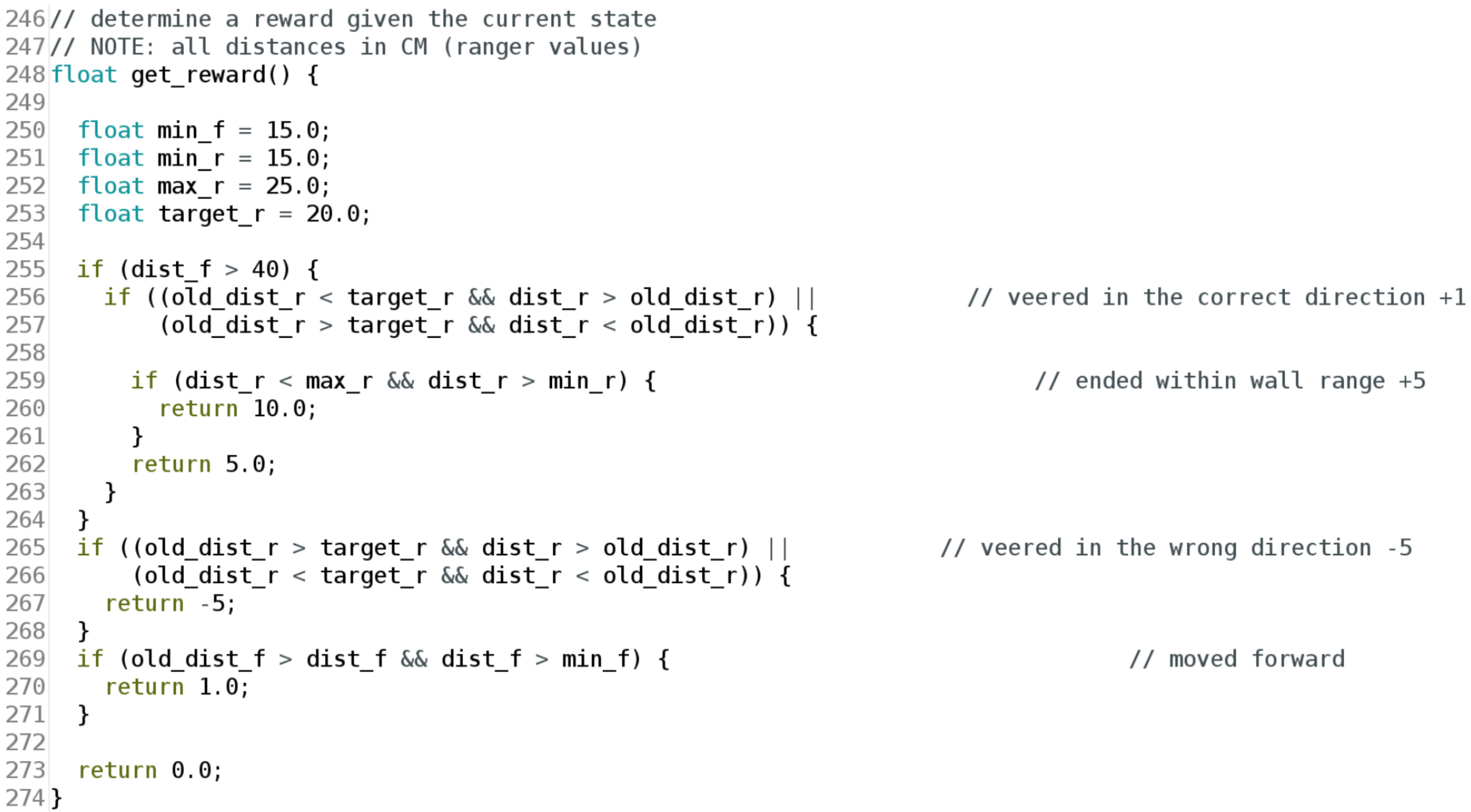
- Different task (wall following)
- Different approximation function
- Different state and action space.

Reinforcement Learning: States



```
// converts a raw state representation to an integer representation
int discretize_state() {
  int root_states = floor(sqrt(NUM_STATES));

  float dist_f_local = clamp(0.0, dist_f, RG_MAX_DIST);
  int norm_f = floor((dist_f_local / RG_MAX_DIST) * (root_states - 1));

  float dist_r_local = clamp(0.0, dist_r, RG_MAX_DIST);
  int norm_r = floor((dist_r_local / RG_MAX_DIST) * (root_states - 1));

  float result = norm_f + root_states * norm_r;

  return result;
}
```

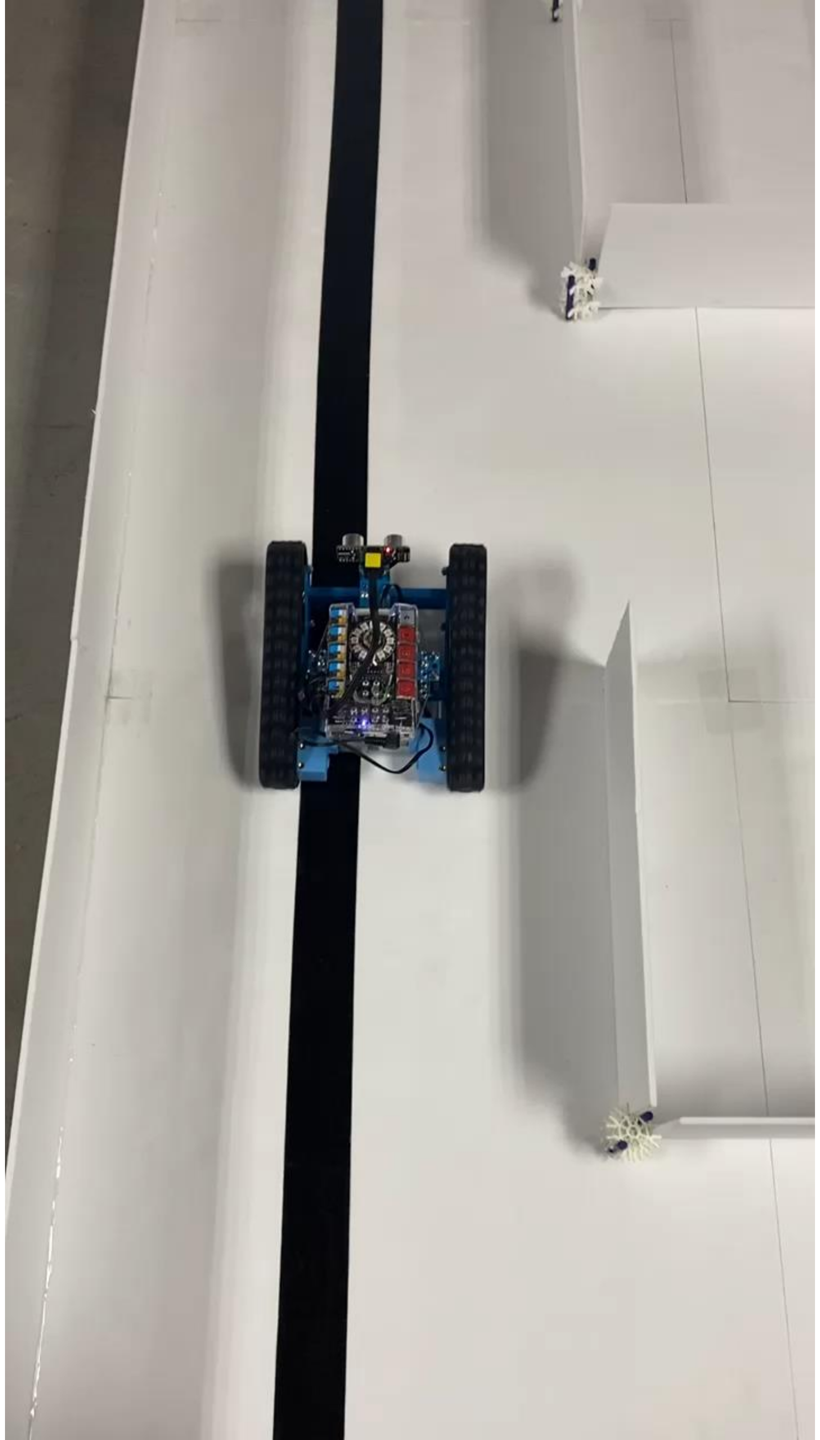
Reinforcement Learning: Rewards



```
// determine a reward given the current state
// NOTE: all distances in CM (ranger values)
float get_reward() {

  float min_f = 15.0;
  float min_r = 15.0;
  float max_r = 25.0;
  float target_r = 20.0;

  if (dist_f > 40) {
    if ((old_dist_r < target_r && dist_r > old_dist_r) ||           // veered in the correct direction +1
        (old_dist_r > target_r && dist_r < old_dist_r)) {

      if (dist_r < max_r && dist_r > min_r) {                            // ended within wall range +5
        return 10.0;
      }
      return 5.0;
    }
  }
  if ((old_dist_r > target_r && dist_r > old_dist_r) ||           // veered in the wrong direction -5
      (old_dist_r < target_r && dist_r < old_dist_r)) {
    return -5;
  }
  if (old_dist_f > dist_f && dist_f > min_f) {                                  // moved forward
    return 1.0;
  }

  return 0.0;
}
```

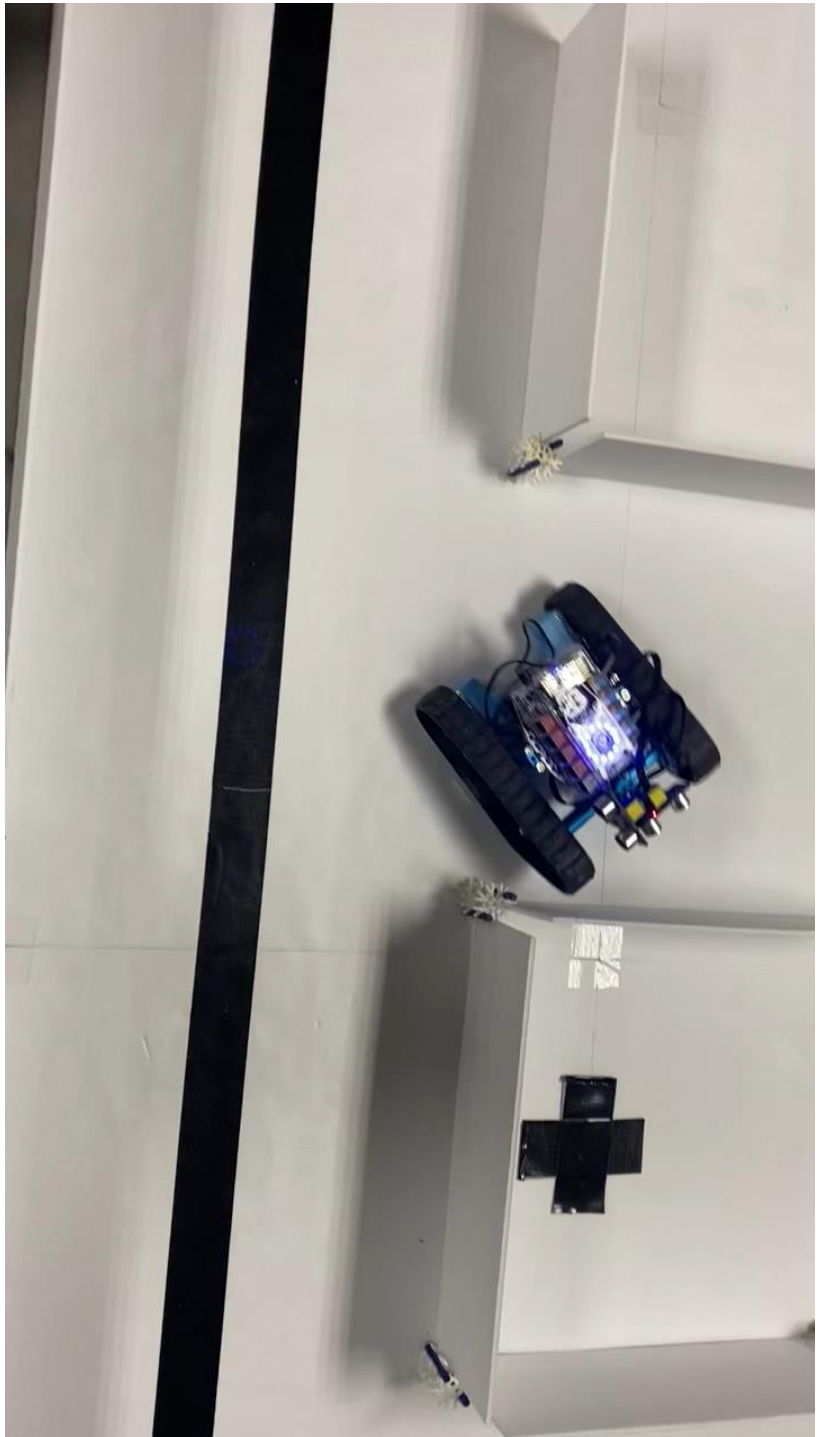
Foreseeable Challenges

- Implementing RL
    - Training phase
- Communication between Bots

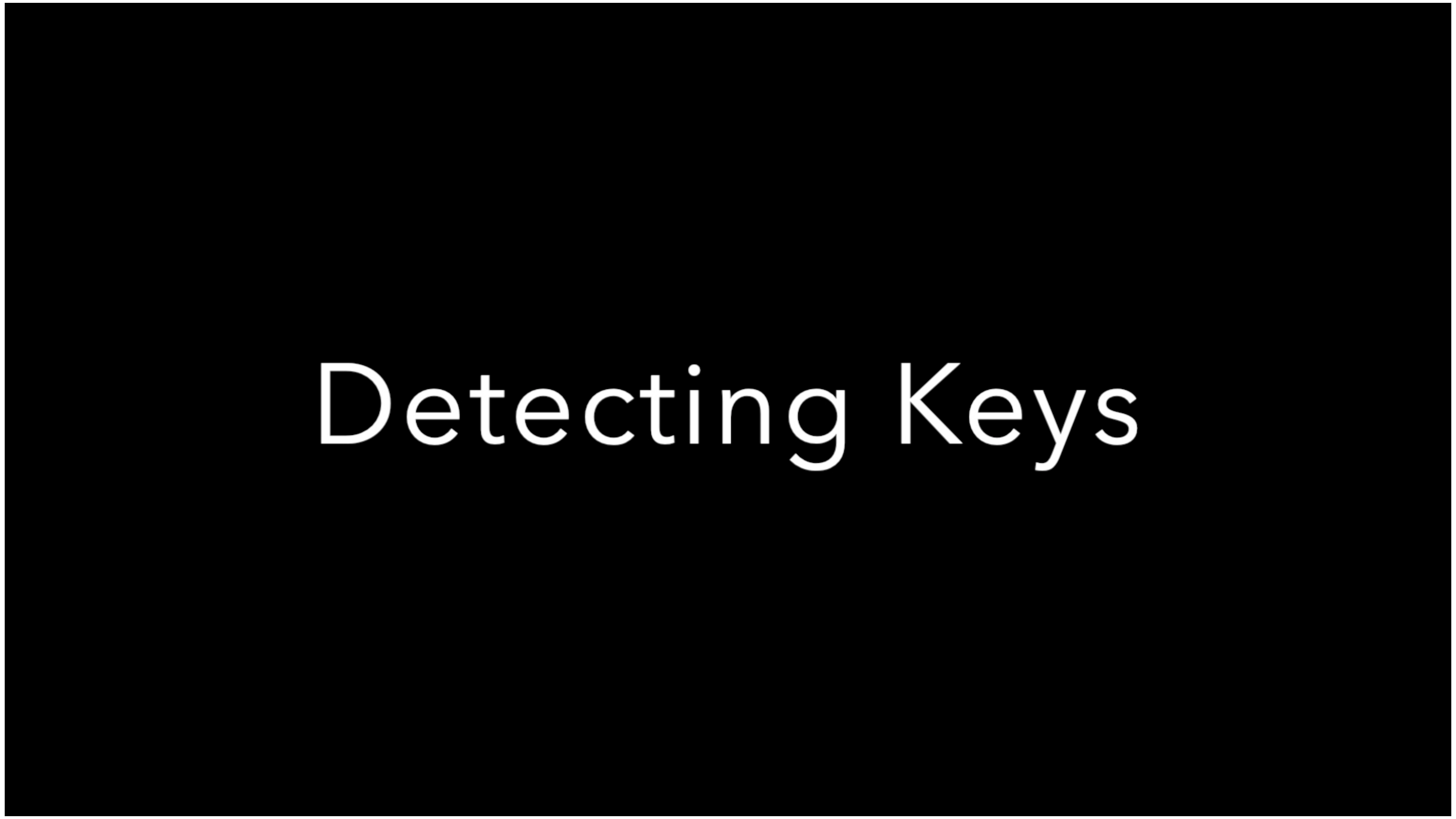


Unexpected Challenges

- Incorporating additional Sensors
- Unexpected states
- Maze Configuration

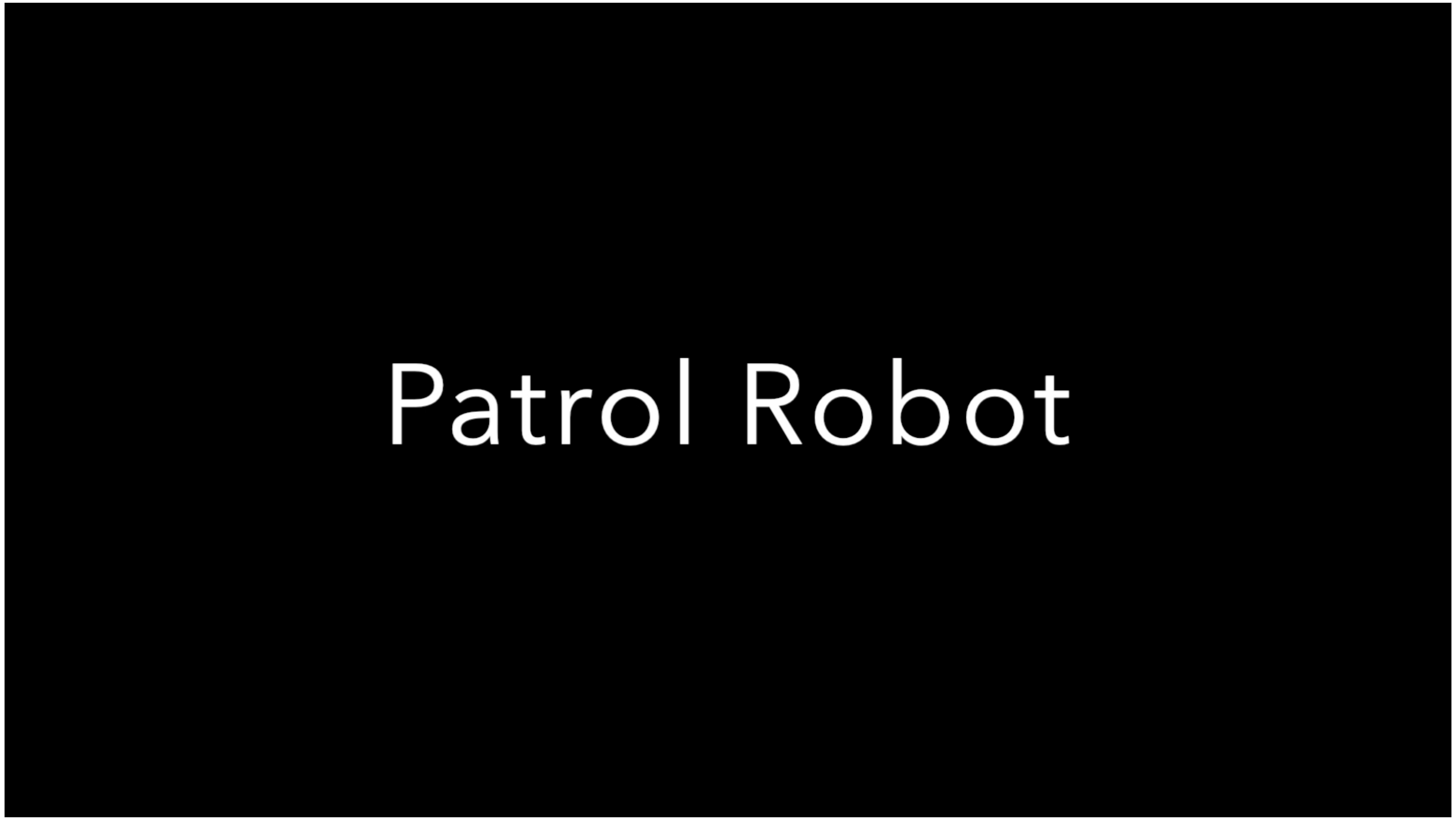


Collecting Keys



Detecting Keys

Patrol Bot



Patrol Robot

Possible Improvements

Deep learning for function approximation

Manual control for training

A physical maze more similar to gazebo

- Thicker walls
- More space in corridors