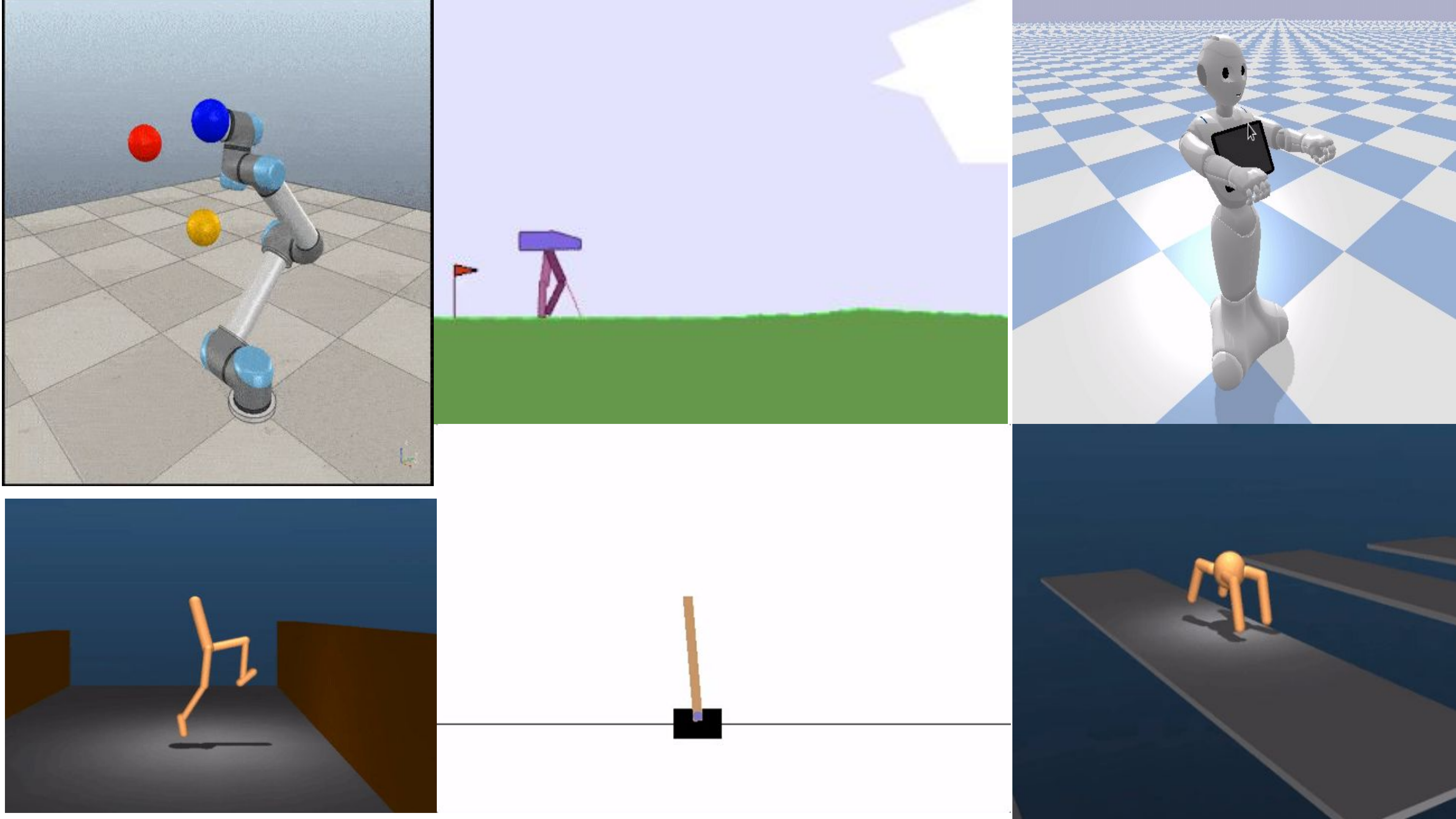
EEE 587 : Optimal Control

OBSTACLE AVOIDANCE USING REINFORCEMENT LEARNING

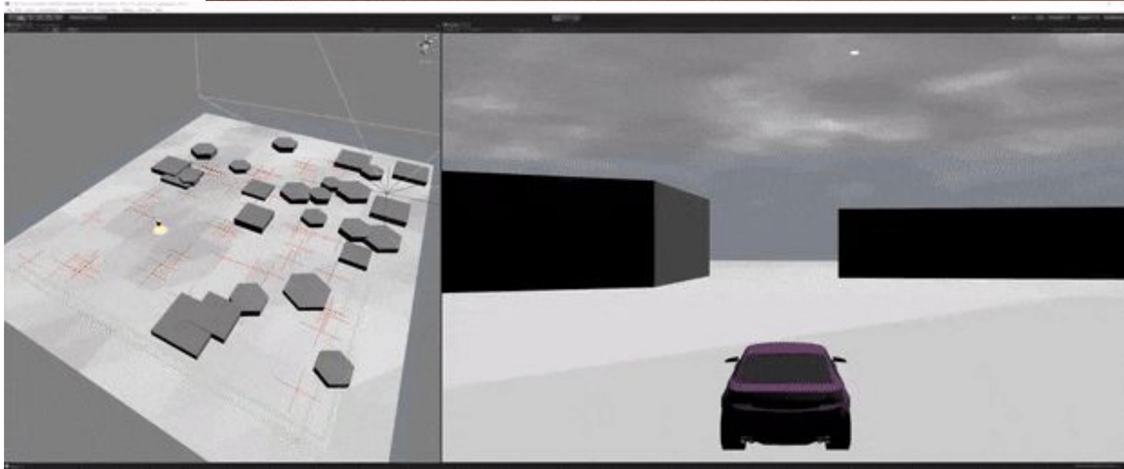
Presenters:
Shravan S Rai
Sivanaga Surya Vamsi Popuri
Varadaraya Ganesh Shenoy

Arizona State University



Obstacle Avoidance Applications

1. Robotic Arms
2. Humanoid
3. Small Mobile Robots
4. Self Driving Cars
5. Satellites



Arizona State University

Abstract

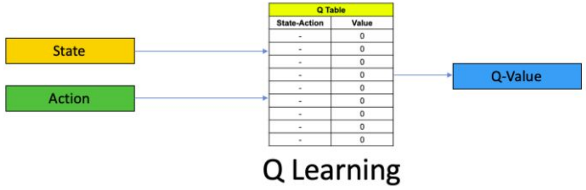
- A new approach to obstacle avoidance generalizes the solution to the problem is by acquiring the local information.
- It is important for the robot to tackle new environments instead of rendering it inoperable.
- Lastly, it is useful to ensure a collision-free trajectory in an ever-changing workspace consisting of static and dynamic obstacles.

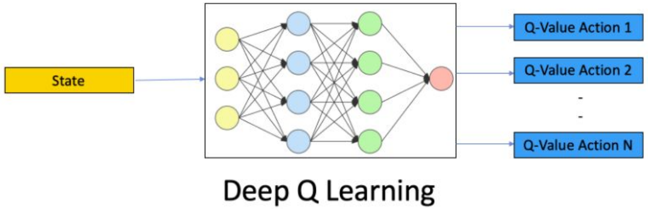
Problem Formulation

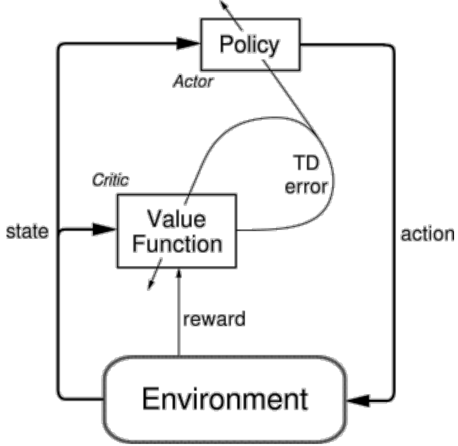
- Navigate from Point A to point B without colliding with the Environment.
- The environment is a grid with walls which act like static obstacles.
- The State of the Robot is given by the Environment.
- Reward the Robot for reaching the Goal location and penalize the Robot for Collision.

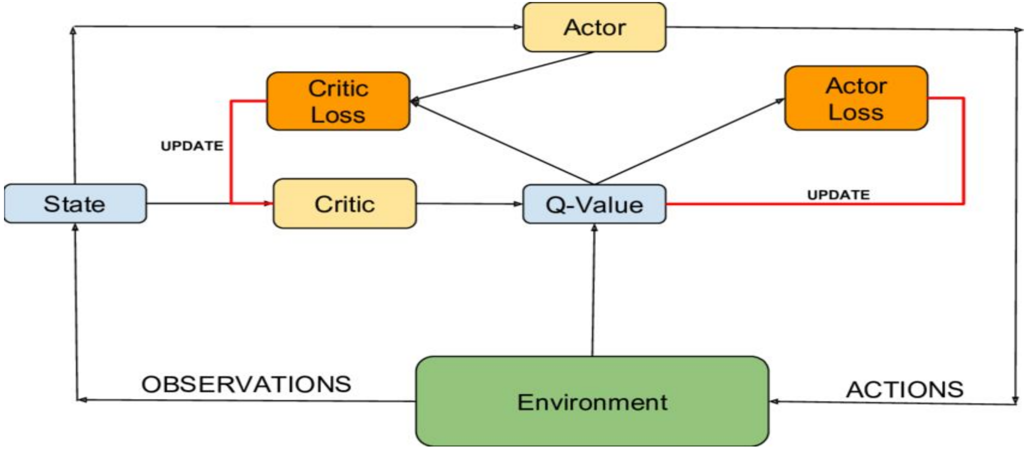
Methods

1. Q Learning
2. Deep Q Learning
3. Actor Critic Method
4. DDPG
5. DDPG with HER



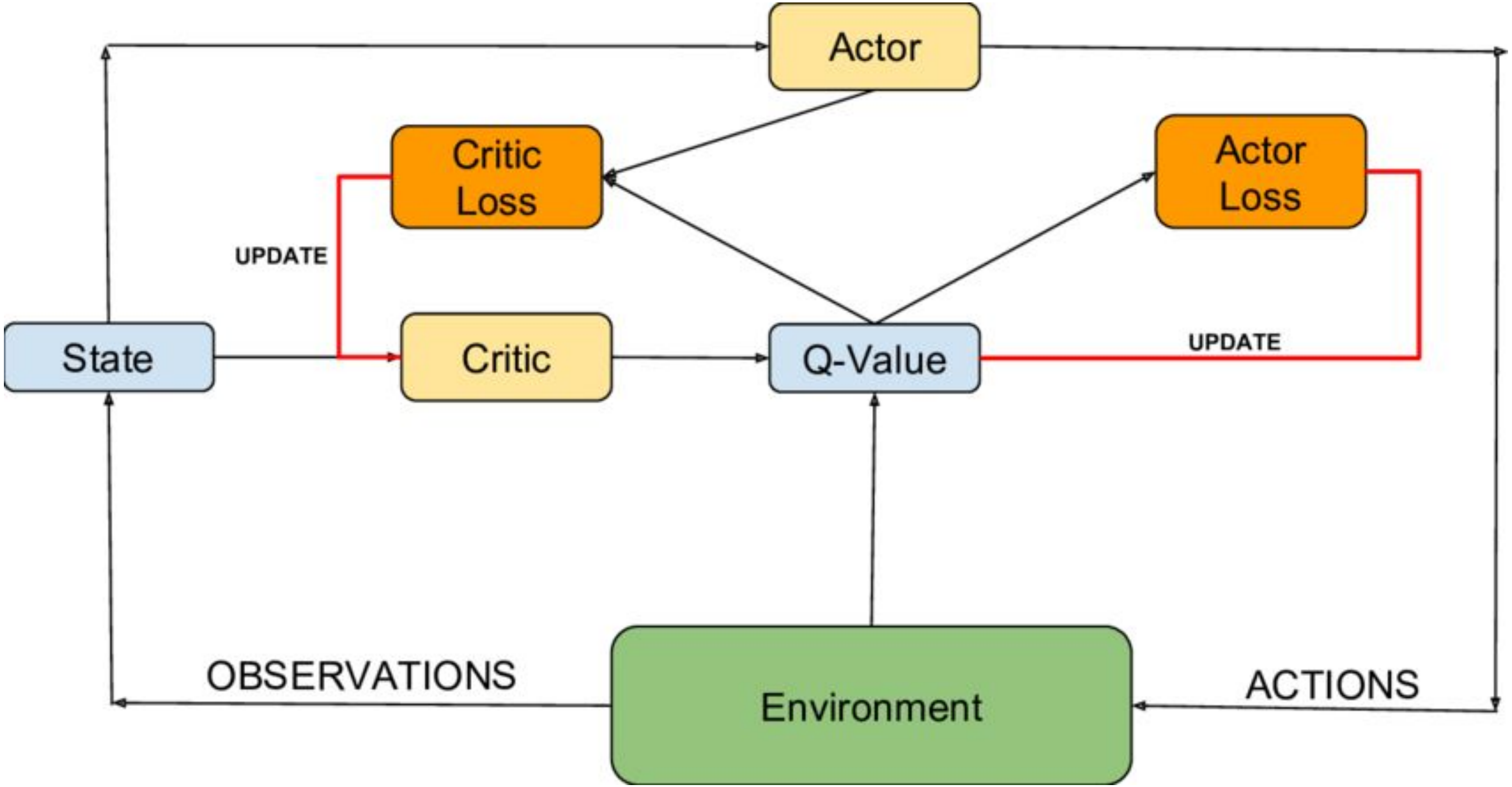






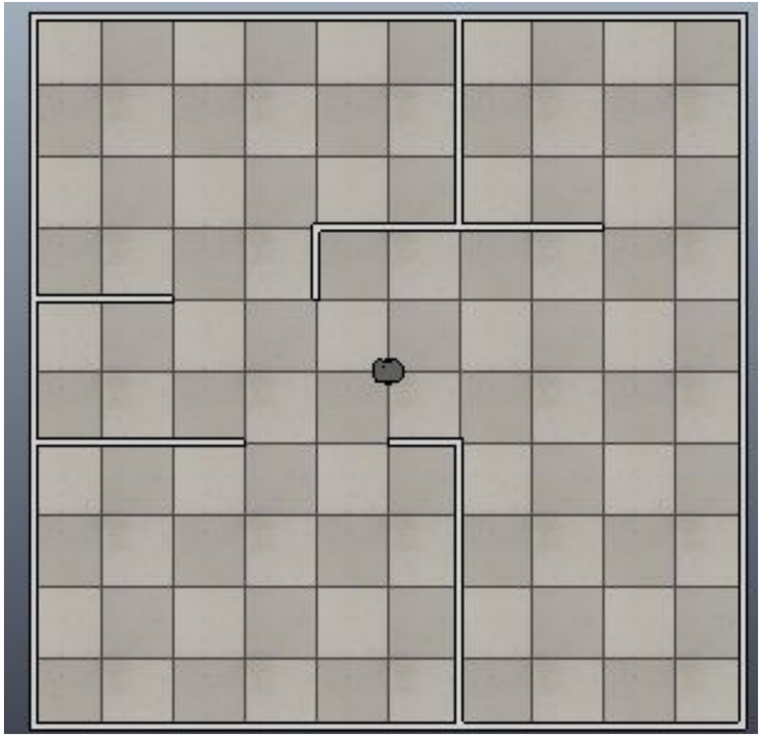
Arizona State University

Deep Deterministic Policy Gradient-DDPG



Implementation

- V-Rep / CoppeliaSim
- PyRep
- Gym-Vrep
- SmartBot
- Ultrasonic Sensors



Arizona State University

Implementation

Observations:

1. 5 distances from Ultrasonic Sensors
2. Distance from the Goal
3. Heading angle of the Bot
4. Linear Velocity of the Robot
5. Angular Velocity of the Robot

Actions:

1. Linear Velocity
2. Angular Velocity

Implementation

Actor Network:

2 Hidden Layers

Layer 1: 200 units, Relu

Layer 2: 100 units, Relu

Output: Linear

Critic Network:

2 Hidden Layers

Layer 1: 200 units, Relu

Layer 2: 100 units, Relu

Output: Linear

Implementation

Reward Function

$$R = \begin{cases} 1 & , d < d_{th} \\ -1 & , \exists D < d_{collision} \\ -0.1 & , \exists D < d_{proxth} \\ V_L \cdot \cos\theta & , otherwise \end{cases}$$

d: distance between robot and goal
d_{th} : threshold distance defined between robot and goal
$d_{collision}$: distance less than which is considered a collision
d_{proxth}: safety distance threshold
D: distance as measured by ultrasonic sensor
V_L: Linear Velocity
θ: Heading angle

Arizona State University

Training

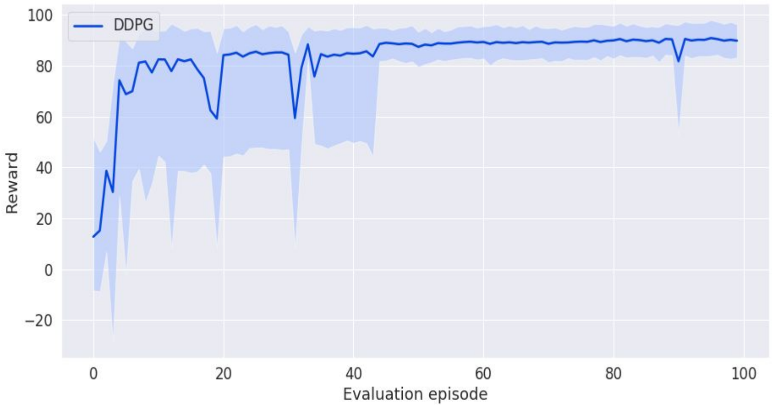
- Number of Episodes: 1000
- Actor Learning Rate: $1e^{-4}$
- Critic Learning Rate: $1e^{-3}$
- 15 start,goal pairs per episode.

```
SPAWN_LIST = np.array([
        [ 0.0,-2.0],
        [ 2.0, 2.0],
        [ 2.0,-2.0],
        [ 2.0, 1.0],
        [ 2.0,-1.0],
        [-1.5,-1.5],
        [-1.0, 2.0],
        [-2.0,-1.0],
        [-2.0, 2.0],
        [-2.0,-2.0],
        [-2.0, 0.0],
        [ 0.0, 2.0],
        [ 0.0, 0.0],
        [ 0.0,-2.0],
        [-1.5,-1.5]
])

GOAL_LIST = np.array([
        [-2.0,-2.0],
        [-1.5,-1.5],
        [-2.0,-1.0],
        [-2.0, 2.0],
        [-1.0, 2.0],
        [ 0.0, 2.0],
        [ 2.0, 2.0],
        [ 2.0, 1.0],
        [ 2.0, 0.0],
        [ 1.0,-0.5],
        [ 2.0,-1.0],
        [ 2.0,-2.0],
        [ 0.0,-2.0],
        [-2.0, 0.0],
        [ 0.0, 0.0]
])
```

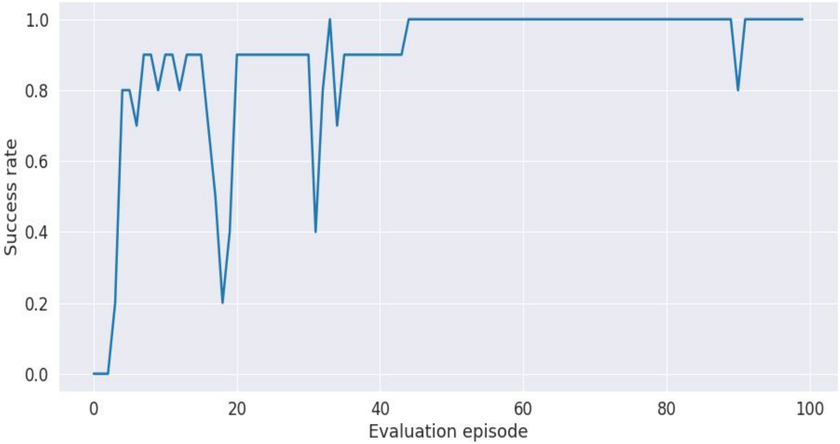
Arizona State University

Results



Plots showing Rewards collected during the 100 episodes of evaluation.

Plots showing Success Rate of reaching the goal location during the 100 episodes of evaluation.



Conclusion

- Successfully Navigate from Start position to Goal Position
- Avoid obstacles successfully hence avoiding collision with the environment.
- Can accommodate random start and goal positions.
- Takes long time to train (approx 2 days), hence not a lot of experiments conducted.