CS4910: Deep Learning for Robotics

David Klee
klee.d@northeastern.edu

T/F, 3:25-5:05pm
Behrakis Room 204

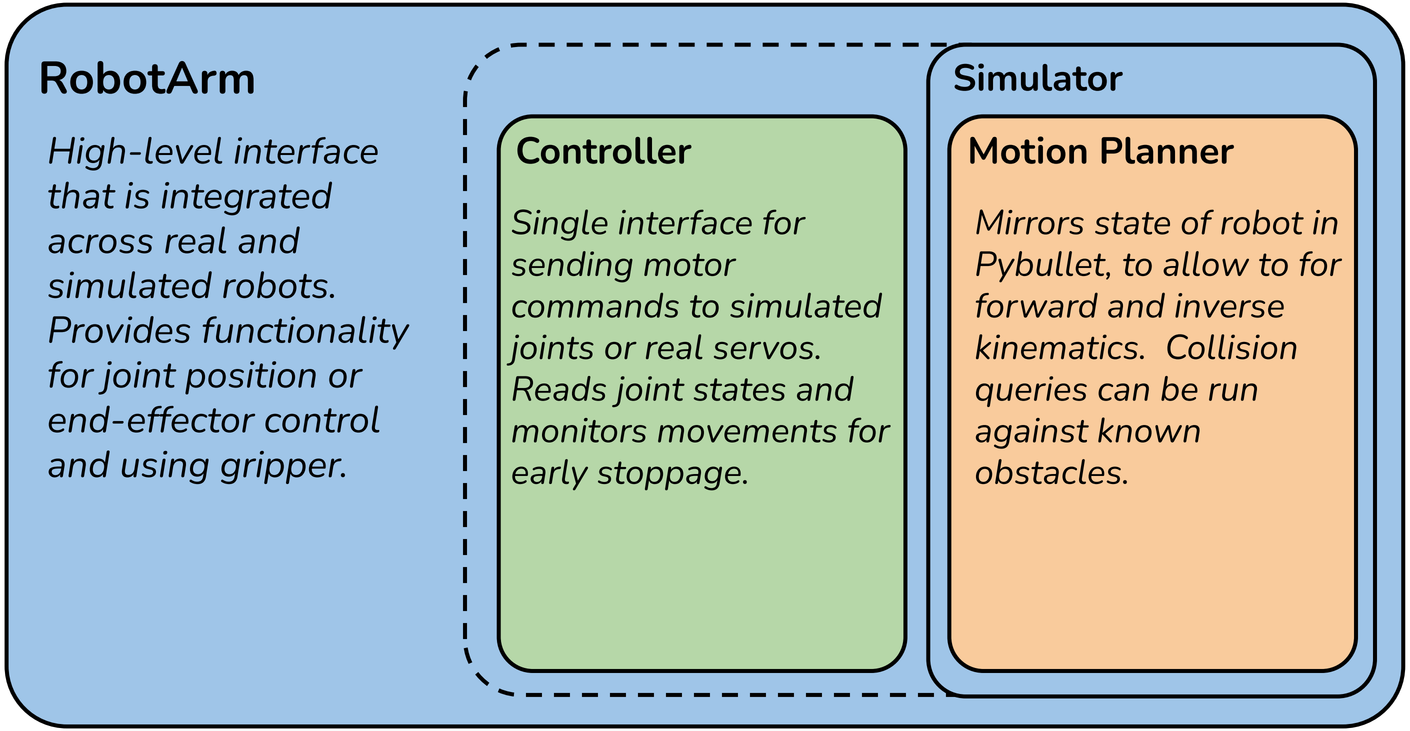
https://www.ccs.neu.edu/home/dmklee/cs4910_s22/index.html

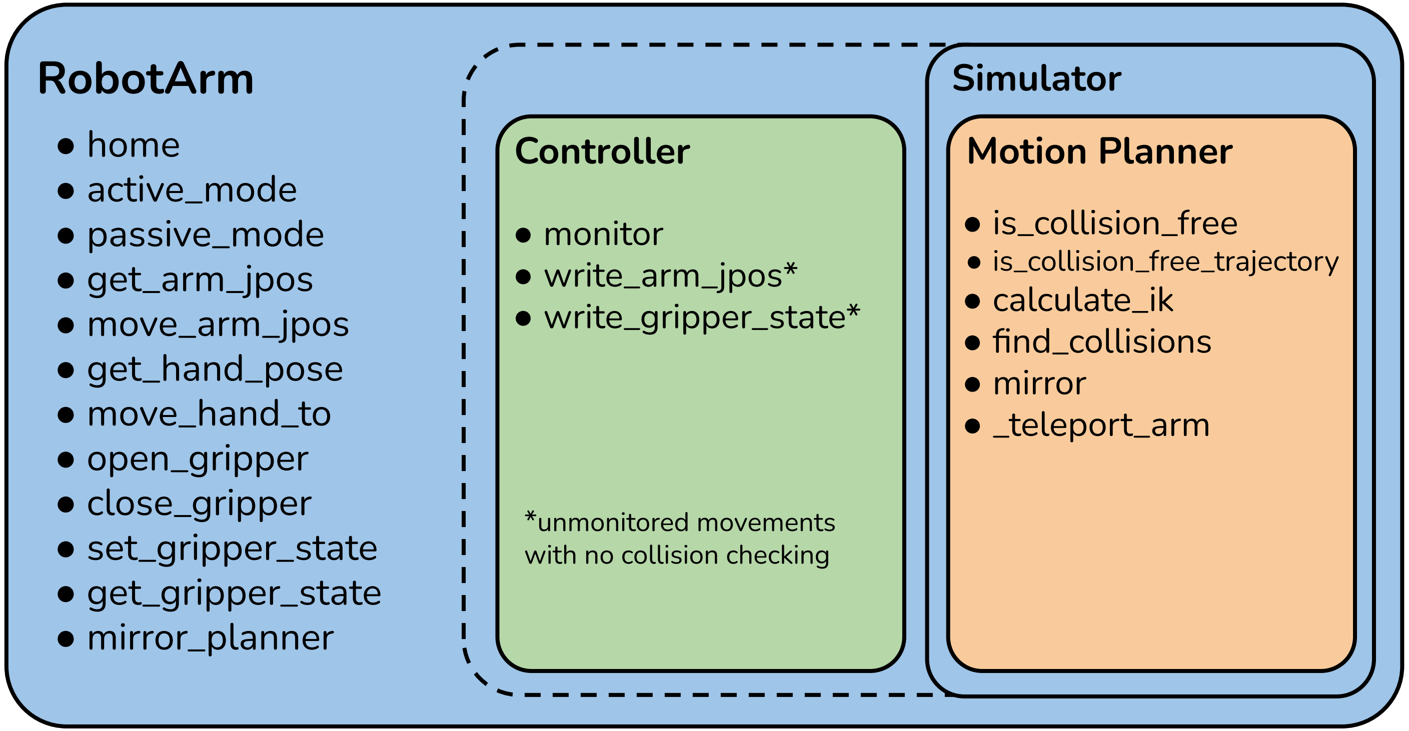
https://piazza.com/northeastern/spring2022/cs4910a/home

Using nuro.arm API

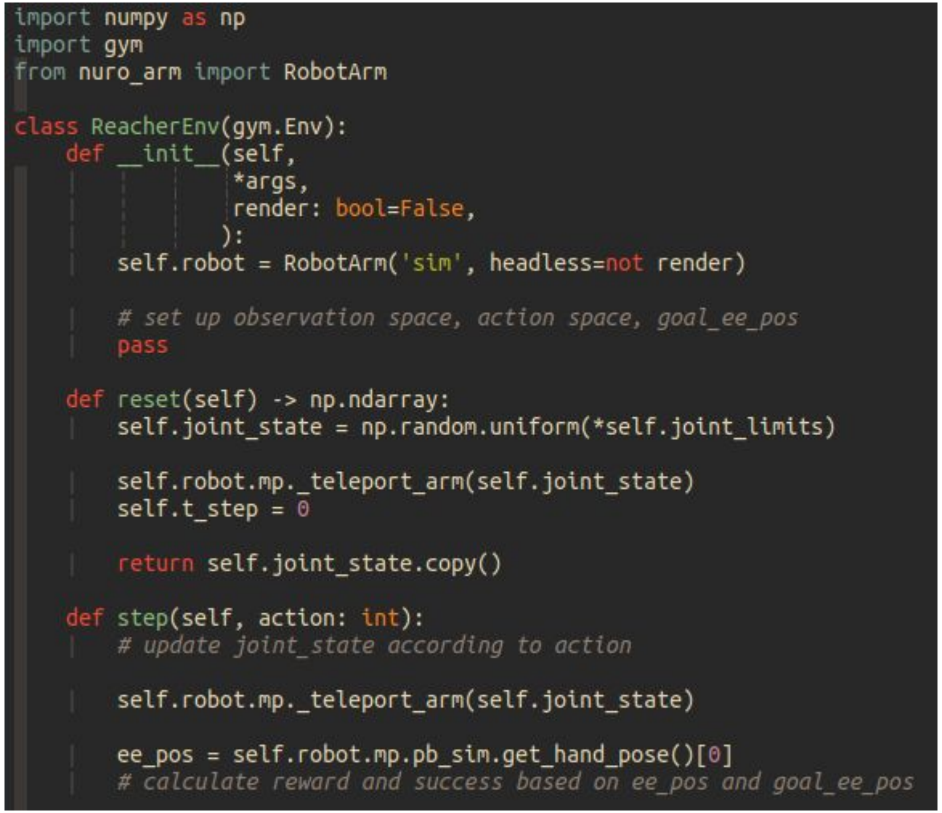
github.com/dmklee/nuro-arm

Overview of **robot** Module


RobotArm
High-level interface that is integrated across real and simulated robots. Provides functionality for joint position or end-effector control and using gripper.
Controller
Single interface for sending motor commands to simulated joints or real servos. Reads joint states and monitors movements for early stoppage.
Simulator
Motion Planner
Mirrors state of robot in Pybullet, to allow to for forward and inverse kinematics. Collision queries can be run against known obstacles.

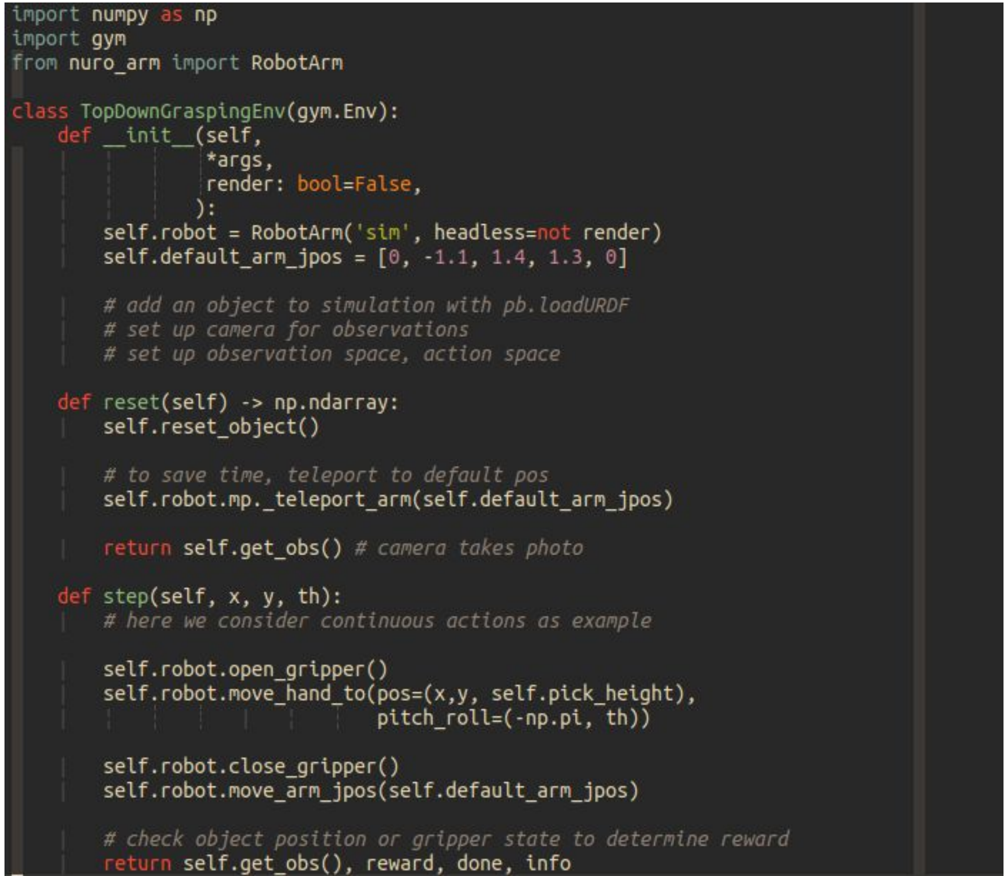

RobotArm
home
active_mode
passive_mode
get_arm_jpos
move_arm_jpos
get_hand_pose
move_hand_to
open_gripper
close_gripper
set_gripper_state
get_gripper_state
mirror_planner
Controller
monitor
write_arm_jpos*
write_gripper_state*
*unmonitored movements with no collision checking
Simulator
Motion Planner
is_collision_free
is_collision_free_trajectory
calculate_ik
find_collisions
mirror
_teleport_arm

For more research-oriented work, you will mostly interact directly with **Controller** and **MotionPlanner**



```python
import numpy as np
import gym
from nuro_arm import RobotArm

class ReacherEnv(gym.Env):
    def __init__(self,
                 *args,
                 render: bool=False,
                ):
        self.robot = RobotArm('sim', headless=not render)

        # set up observation space, action space, goal_ee_pos
        pass

    def reset(self) -> np.ndarray:
        self.joint_state = np.random.uniform(*self.joint_limits)

        self.robot.mp._teleport_arm(self.joint_state)
        self.t_step = 0

        return self.joint_state.copy()

    def step(self, action: int):
        # update joint_state according to action

        self.robot.mp._teleport_arm(self.joint_state)

        ee_pos = self.robot.mp.pb_sim.get_hand_pose()[0]
        # calculate reward and success based on ee_pos and goal_ee_pos
```

For more research-oriented work, you will mostly interact directly with **Controller** and **MotionPlanner**



```python
import numpy as np
import gym
from nuro_arm import RobotArm

class TopDownGraspingEnv(gym.Env):
    def __init__(self,
                 *args,
                 render: bool=False,
                ):
        self.robot = RobotArm('sim', headless=not render)
        self.default_arm_jpos = [0, -1.1, 1.4, 1.3, 0]

        # add an object to simulation with pb.loadURDF
        # set up camera for observations
        # set up observation space, action space

    def reset(self) -> np.ndarray:
        self.reset_object()

        # to save time, teleport to default pos
        self.robot.mp._teleport_arm(self.default_arm_jpos)

        return self.get_obs() # camera takes photo

    def step(self, x, y, th):
        # here we consider continuous actions as example

        self.robot.open_gripper()
        self.robot.move_hand_to(pos=(x,y, self.pick_height),
                                pitch_roll=(-np.pi, th))

        self.robot.close_gripper()
        self.robot.move_arm_jpos(self.default_arm_jpos)

        # check object position or gripper state to determine reward
        return self.get_obs(), reward, done, info
```

Example scripts in nuro-arm repo

1. Mirroring robot in simulator

```
$ python nuro_arm/examples/real_sim_mirroring.py
```

2. Record movements

```
$ python nuro_arm/examples/record_movements.py
```

3. Complex motion*

```
$ python nuro_arm/examples/complex_motion.py
```

**can be done with simulator only (--sim)*

Camera module

- Calibration to calculate intrinsic and extrinsic parameters
- Aruco Tag localization
- Support for replicating real camera’s images in simulator

Getting feedback

Please let me know if there are bugs or organizational issues that you notice with the API, I am happy to talk about possible fixes

On the horizon

Friday:

- HW3 questions and UNet implementation
- Discussion of Project

Next Tuesday:

- Quick Project Pitches
- Example Project Walk through

Next Friday:

- Special Topic Lecture

Survey to provide feedback

https://forms.gle/AFCKEZuiRxpvFAYP7