`RTPlayground`

November 23, 2022

1 Background and Motivation

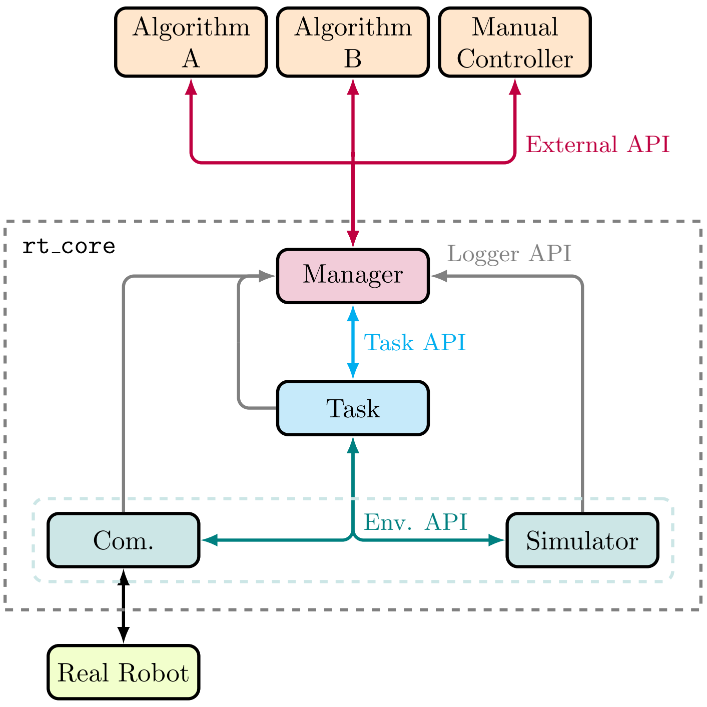
Training, testing, and comparing RL algorithms require controlled experiments and environmental setup. Unfortunately, there is a lack of uniformity and standardization when it comes to RL training and testing on real world tasks. **`RTPlayground` (Robot Training Playground) aims to provide a standardized framework for testing RL algorithms on both simulated and real world experiments**.

Our objective is to provide users with:

1. A standardized API for setting up environments and tasks; logging metrics on both simulated and real world experiments; and environment rollouts on both simulations and real world environments.
2. The ability to easily compare results between different algorithms and between simulations and real world experiments.

2 Conceptual Design

The diagram below illustrates the connection between the various components of the framework.



2.1 `rt_core`: Components and APIs

- **Manager Module**: Orchestrates the flow of information from the task and environment modules to the external modules. The manager also provides an interface for a centralized logging system that allow comparison between multiple algorithms

- `RTPlayground` focuses on Markov decision processes (MDP), defined with a 4-tuple (S, A, T, R) where S and A are sets of states and actions, T is the transition probability, and R is the reward function. The following components enables the development of various MDPs for training and testing:
  - **Task Module**: Configure and output the reward function $r = R(s, a)$, task termination criteria $d = f_{\text{done}}(s, a)$, and initial state probabilities $s_0 \sim S_0$.
  - **Environment**: The "environment" consists of the robot and its environment. The environment is the source of the state transition probability $s' \sim T(s, a)$ and the state-observation mapping $o \sim O(s)$. In order to enable both real world and simulated experiments, the environment module is broken down into two separate modules as follows:
    - **Communicator Module**: Communication interface with the real robot.
    - **Simulator Module**: Simulates the robot and provides a standardized communication interface with the simulated robot.
- **External API**: An outward facing API, based on OpenAI Gym's API.
- **Task API**: A standardized API for querying and setting up task modules.
- **Environment API**: A standardized API for interacting and setting up environment modules.

2.2 External Modules

- **Training algorithms**
- **Testing algorithms**
- **Data collecting algorithm**: For offline RL, imitation learning, etc.
- **Manual control modules**: For manually controlling robot via a user interface.