Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №12
по дисциплине ««Основы систем ИИ»

Тема Подкрепление

Студент Батуев А.Г.

Группа ИУ7-36Б

Преподаватели Строганов Ю.В.

Москва, 2025

Содержание

ВВЕДЕНИЕ

Целью лабораторной работы является разработка алгоритма управления роботизированной кистью Adroit для выполнения задачи Adroit Hammer с использованием методов обучения с подкреплением. Объектом исследования выступает виртуальная модель роботизированной кисти Adroit Hand в среде симуляции Gymnasium-robotics. Модель кисти включает в себя лучезапястный сустав, суставы пальцев и 24 степени свободы, управляемые вектором действий, содержащим значения углов поворота в каждом суставе. Задача Adroit Hammer предполагает манипулирование молотком для достижения заданной целевой позиции гвоздя (гвоздь забит).

Для достижения поставленной цели необходимо выполнить следующие задачи:

1) выбрать и реализовать алгоритм обучения с подкреплением для управления кистью;
2) обучить агента взаимодействовать со средой и выполнять задачу Adroit Hammer;
3) проанализировать результаты обучения и осуществить оценку качества выполнения задачи.

1 Аналитическая часть

1.1 Обучение с подкреплением

Обучение с подкреплением (reinforcement learning, RL) представляет собой область машинного обучения, в которой агент учится действовать в среде с целью максимизации накопленной награды [1]. Основными компонентами системы RL являются:

— агент – обучаемая система, принимающая решения;

— среда – окружение, с которым взаимодействует агент;

— награда – числовая величина, определяющая успех действия агента;

— состояние – описание текущего положения среды;

— действие – выбор агента, влияющий на среду.

Процесс обучения состоит из взаимодействия агента со средой, при котором агент наблюдает текущее состояние, выбирает действие, получает награду и новое состояние. Этот процесс можно представить как задачу оптимизации [1], где цель — максимизация ожидаемой совокупной награды, задаваемой следующим образом:

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}, \tag{1.1}$$

где G_t — ожидаемая совокупная награда с момента времени t, $\gamma \in [0, 1]$ — коэффициент дисконтирования, R_{t+k+1} — награда, полученная на шаге $t + k + 1$.

Существует множество алгоритмов обучения с подкреплением, которые можно разделить на три основные группы: методы обучения политики, методы обучения ценности и гибридные методы.

Политика (обычно обозначается π) определяет стратегию агента по выбору действий. Формально она представляется как отображение состояний в действия (или в распределения вероятностей по действиям).

Ценность (функции ценности) оценивают желательность пребывания в определенном состоянии или выполнения конкретного действия. В частности, функции ценности состояния оценивают ожидаемую кумулятивную награду, начиная с состояния s и следуя определенной политике.

1.2 Proximal Policy Optimization (PPO)

Proximal Policy Optimization (PPO) — это метод обучения политики, который улучшает стабильность и эффективность обучения за счет ограничения изменения политики между обновлениями [2]. Цель PPO – оптимизировать стратегию, максимизируя ожидаемое вознаграждение, при этом ограничивая изменения стратегии на каждом шаге обучения, чтобы обеспечить стабильность обучения.

Основная идея PPO заключается в максимизации целевой функции, которая записыва-

ется следующим образом:

$$L^{CLIP}(\theta) = \mathbb{E}_t \left[\min \left(r_t(\theta) A_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon) A_t \right) \right], \tag{1.2}$$

где $\mathbb{E}_t$ – математическое ожидание по временным шагам t, A_t – оценка преимущества (advantage) в момент времени t которая показывает, насколько действие a_t в состоянии s_t лучше или хуже среднего ожидаемого действия в этом состоянии, ϵ – гиперпараметр, ограничивающий изменение политики, $\text{clip}(...)$ – функция, обрезающая значение $r_t(\theta)$, чтобы отношение вероятностей не уходило слишком далеко от единицы, $\min(...)$ – функция минимума, ограничивающая обновление политики снизу, предотвращая слишком пессимистичные обновления в случаях, когда $A_t < 0$.

Отдельно стоит отметить функцию:

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \tag{1.3}$$

где $r_t(\theta)$ – показывает, насколько вероятнее стало действие при новой политике по сравнению со старой, $\pi_\theta(a_t|s_t)$ – вероятность совершить действие a_t в состоянии s_t в момент времени t в соответствии с текущей политикой, определяемой параметрами θ, $\pi_{\theta_{old}}(a_t|s_t)$ – то же самое но для старой политики, определяемой параметрами θ_{old}.

1.3 Soft Actor-Critic (SAC)

Soft Actor-Critic (SAC) – это алгоритм обучения с подкреплением, основанный на идее максимизации ожидаемой совокупной награды с одновременным увеличением энтропии политики [2]. Агент стремится не только получать высокие награды, но и сохранять высокую степень случайности в своих действиях, что способствует исследованию среды и предотвращает преждевременную сходимость к субоптимальным решениям.

Функция потерь записывается следующим образом:

$$L_{SAC}(\pi) = \mathbb{E}_{s_t \sim D, a_t \sim \pi} \left[Q(s_t, a_t) - \alpha \log \pi(a_t|s_t) \right], \tag{1.4}$$

где $\mathbb{E}_{s_t \sim D, a_t \sim \pi}$ – математическое ожидание, где состояние s_t в момент времени t берется из буфера воспроизведения D, а действие a_t в момент времени берется из политики π, $Q(s_t, a_t)$ — функция ценности действия оценивающая ожидаемую совокупную награду при совершении действия, α — гиперпараметр, определяющий влияние энтропии, $\log \pi(a_t|s_t)$ — энтропия политики.

1.4 Deep Deterministic Policy Gradient (DDPG)

Deep Deterministic Policy Gradient (DDPG) — это метод обучения с подкреплением, использующий детерминированную политику и обучающий ее с помощью градиентного спуска

[2]. Целевая функция для обновления политики записывается следующим образом:

$$L_{DDPG}(\theta) = \mathbb{E}_{s_t \sim D}\left[\nabla_\theta \pi_\theta(s_t) \nabla_a Q(s_t, a)|_{a=\pi_\theta(s_t)}\right], \tag{1.5}$$

где $\mathbb{E}_{s_t \sim D}$ – математическое ожидание, где состояние s_t берется из буфера воспроизведения D, $\pi_\theta(s_t)$ – детерминированная политика агента, $\nabla_\theta \pi_\theta(s_t)$ – градиент политики по ее параметрам θ в состоянии s_t, $Q(s_t, a)$ — функция ценности действия, оценивающая ожидаемую совокупную награду, $\nabla_a Q(s_t, a)|_{a=\pi_\theta(s_t)}$ – градиент Q-функции, вычисленный в точке $a = \pi_\theta(s_t)$.

Все выражение представляет собой цепочку градиентов, вычисленное по правилу цепочки (chain rule). Оно показывает, как нужно изменить параметры политики θ, чтобы увеличить значение Q-функции, то есть, чтобы политика выдавала действия, ведущие к большей награде.

1.5 Twin Delayed Deep Deterministic Policy Gradient (TD3)

Twin Delayed Deep Deterministic Policy Gradient (TD3) — это улучшение метода DDPG, направленное на устранение переоценки функции ценности [2]. Основные усовершенствования включают использование двух критиков и добавление шума для уменьшения переоценок. Целевая функция TD3 записывается следующим образом:

$$L_{TD3}(\theta_i) = \mathbb{E}s_t, a_t, r_t, st+1 \sim D\left[(y_t - Q_{\theta_i}(s_t, a_t))^2\right] \quad \text{for } i = 1, 2, \tag{1.6}$$

где $\mathbb{E}s_t, a_t, r_t, st+1 \sim D$ – математическое ожидание по переходам $s_t, a_t, r_t, st+1$ из буфера воспроизведения D, $s_t, a_t, r_t, st+1$ – соответственно состояние, действие, награда и следующее состояние, $Q_{\theta_i}(s_t, a_t)$ – функция ценности действия для i-го критика для состояния s_t и действия a_t, y_t – целевое значение.

Целевое значение вычисляется слудющим образом:

$$y_t = r_t + \gamma \min_{i=1,2} Q_{\theta_i'}(s_{t+1}, \pi_{\phi'}(s_{t+1}) + \epsilon), \tag{1.7}$$

где r_t - полученная награда, γ - коэффициент дисконтирования (обычно 0.95-0.99), $Q_{\theta_i'}(s_{t+1}, a)$ - целевые Q-функции (target Q-networks) с "замороженными"параметрами θ_i', $\pi_{\phi'}(s_{t+1})$ - целевая политика, ϵ - шум, добавляемый к действию целевой политики для сглаживания, $\min_{i=1,2} Q_{\theta_i'}(s_{t+1}, ...)$ - минимум из значений двух целевых Q-функций, что и является основным нововведением TD3 для борьбы с переоценкой.

Вывод

В данной аналитической части были рассмотрены основные алгоритмы обучения с подкреплением, их особенности и области применения. Учитывая сложность задачи управления роботизированной рукой, а также требования к стабильности, универсальности и простоте реализации, PPO представляется наиболее подходящим выбором на начальном этапе исследования.

2 Конструкторская часть

2.1 Процесс обучения

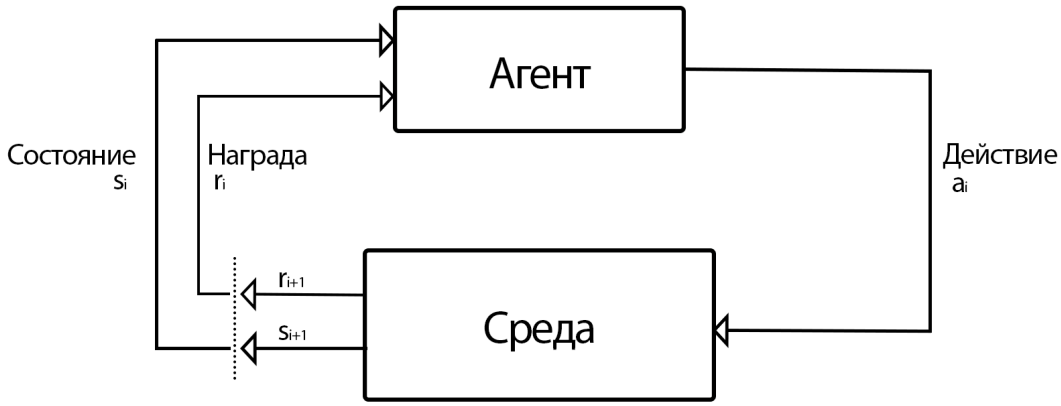


Рисунок 2.1 — Цикл обучения с подкреплением

Процесс обучения начинается с получения информации о состоянии s_t из среды. Основываясь на этом состоянии, агент выполняет дейсвтие a_t, тем самым среда перемещается в новое состояние s_{t+1}. Среда даёт вознаграждение агенту r_{t+1}. Агент использует вознаграждение как сигнал для корректировки действий, стремясь максимизировать награду.

2.2 Описание среды AdroitHandHammer-v1

AdroitHandHammer-v1 — это среда для обучения с подкреплением, являющаяся частью библиотеки $\mathrm{gymnasium\_robotics}$ и основанная на симуляторе MuJoCo [3].

Среда предоставляет возможности:

— наблюдения за:

– положением и ориентацией суставов роботизированной руки, задаваемыми через 24 степени свободы, каждая из которых представлена тремя значениями (синус, косинус, угловая скорость), что в сумме дает 72 вещественных значения;

– положением молотка в трёхмерном пространстве, определяемым 3 вещественными значениями (x, y, z);

– положением гвоздя в трёхмерном пространстве, определяемым 3 вещественными значениями (x, y, z);

– линейной и угловой скоростью молотка, определяемыми 3 вещественными значениями (x, y, z) для каждой компоненты;

– расстоянием от головки молотка до шляпки гвоздя, определяемым как вещественная разность координат (x, y, z);

– нормализованным вектором, указывающим направление от головки молотка до шляпки гвоздя, определяемым вещественными значениями (x, y, z) в интервале от -1 до 1;

– состоянием гвоздя относительно доски, которое задаётся 2 вещественными значениями от 0 (гвоздь не забит) до 1 (гвоздь полностью забит);

— управлением 24 степенями свободы роботизированной руки, что позволяет задавать крутящий момент для каждого сустава;

— регулированием вознаграждением:

– агент получает награду только если гвоздь полностью забит (разряженная награда);

– агент получает награду, основанную на расстоянии между головкой молотка и шляпкой гвоздя, а также на прогрессе забивания гвоздя (плотная награда);

— установлением условия завершения: эпизод завершается, если гвоздь полностью забит или по истечении максимального количества шагов;

— установкой начального состояния, при котором роботизированная рука находится в исходном положении, молоток лежит на столе, а гвоздь частично вбит в доску;

— сбором данных, полученных с помощью различных политик (в том числе экспертных и случайных).

2.3 Общий ход алгоритма

Процесс обучения модели машинного обучения можно разделить на несколько основных этапов: инициализацию, тренировку, тестирование и визуализацию результатов.

На этапе инициализации подготавливается все необходимое для обучения и тестирования: среда, модель, инструменты для отслеживания процесса обучения.

Тренировка (обучение) является основным этапом, на котором модель подстраивает свои параметры для лучшего выполнения задачи. Сначала запускается процесс обучения модели на определенное количество шагов. Во время обучения модель взаимодействует со средой следующим образом: она получает данные о текущем состоянии среды (наблюдение), выбирает действие на основе своей текущей стратегии, среда изменяет свое состояние в соответствии с действием и выдает награду, а модель обновляет свою стратегию, стремясь максимизировать суммарную награду. По завершении обучения модель сохраняется для дальнейшего использования.

На этапе тестирования обученная модель проверяется на способность решать поставленную задачу. Сначала среда возвращается в начальное состояние. Затем модель предсказывает действие на основе текущего состояния, среда выполняет шаг, используя предсказанное действие, и возвращает новое состояние, награду и информацию о завершении эпизода. Цикл

взаимодействия со средой повторяется до завершения эпизода.

Наконец, происходит визуализация результатов. Строится и сохраняется график, отображающий изменение показателей в процессе обучения (например, сумму наград). Это позволяет визуально оценить эффективность обучения.

Вывод

В данной главе были рассмотрены ключевые аспекты процесса обучения с подкреплением на примере среды AdroitHandHammer-v1. Описан цикл взаимодействия агента со средой, включающий получение наблюдений, выполнение действий и получение вознаграждений. Подробно описана среда AdroitHandHammer-v1, включая ее возможности, структуру наблюдений, действий, наград, а также условия завершения эпизодов и начальное состояние. Также был изложен общий алгоритм процесса обучения модели, состоящий из этапов инициализации, тренировки, тестирования и визуализации. Таким образом, была подготовлена база для дальнейшего построения и обучения модели, способной успешно решать задачу забивания гвоздя в среде AdroitHandHammer-v1.

3 Технологическая часть

3.1 Выбор языка и среды программирования

Для реализации программного обеспечения были использованы следующие средства:

— среда разработки PyCharm 2023 community edition [4];

— язык разработки Python 3.12 [5].

Выбор языка программирования Python 3.12 обоснован наличием все необходимых библиотек для выполнения поставленных задач. Среда разработки Pycharm 2023 community edition выбрана для написания и отладки кода.

Используемые библиотеки:

— gymnasium — предоставляющая интерфейс для работы со средой [6];

— gymnasium_robotics — предоставляющая среду AdroitHandHammer-v1 [3];

— stable_baselines3 — библиотека для реализации алгоритмов машинного обучения, в частности PPO [7];

— numpy — для быстрой работы с массивами, сохранения промежуточных данных и связи их с другими библиотеками [8];

— matplotlib.pyplot — для отображения графиков [9];

— pathlib.Path — доступ к файлам системы [10].

3.2 Программная реализация

В листинге 3.1 представлен код, которыи выполняет подготовительные действия для обучения модели с подкреплением.

Листинг 3.1 — Инициализация

```
import gymnasium as gym
import gymnasium_robotics
from gymnasium.wrappers import RecordVideo
import matplotlib.pyplot as plt
import numpy as np
from pathlib import Path
from stable_baselines3 import PPO
from stable_baselines3.common.callbacks import BaseCallback
WORK_DIR = Path.cwd().resolve()
IMG_DIR = WORK_DIR / "report" / "images"
VID_DIR = WORK_DIR / "video"
env = gym.make('AdroitHandHammer-v1', render_mode="rgb_array")
env = RecordVideo(env, str(VID_DIR), episode_trigger=lambda e: True)
device = "cuda" if torch.cuda.is_available() else "cpu"
```

Сначала подключаются необходимые инструменты. Затем определяются пути к рабочим

директориям, где будут сохраняться изображения и видео. Создается среда обучения под названием AdroitHandHammer-v1, в которой агент будет управлять роботизированной рукой, чтобы забить гвоздь молотком. Среда настраивается таким образом, чтобы можно было получать визуальное представление происходящего в виде изображения. Дополнительно включается запись видео, причем записываться будет каждый эпизод взаимодействия агента со средой. Наконец, определяется, какое устройство будет использоваться для вычислений: графический процессор, если он доступен, или центральный процессор в противном случае.

В листинге 3.2 представлен код, который выполняет обучение модели с подкреплением и настройку обратного вызова.

Листинг 3.2 — Обучение модели и настройки колбэка

```
class TrainingMonitorCallback(BaseCallback):
    def __init__(self, verbose=0):
        super().__init__(verbose)
        self.rewards = []
    def _on_step(self) -> bool:
        if "rewards" in self.locals:
            self.rewards.append(self.locals["rewards"])
        return True
model = PPO("MlpPolicy", env, verbose=1, device=device)
callback = TrainingMonitorCallback()
total_timesteps = 1000000
model.learn(total_timesteps=total_timesteps, callback=callback)
model.save("ppo_adroit_hammer")
```

Создается обратный вызов, предназначенная для сбора данных о наградах (rewards) в процессе обучения. Модель PPO с политикой *MlpPolicy* обучается в среде env в течение 1 000 000 временных шагов. Колбэк *TrainingMonitorCallback* добавляется в процесс обучения для мониторинга. На каждом шаге обучения колбэк проверяет наличие переменной *rewards* в локальных переменных и, если она существует, добавляет ее значение в список. После завершения обучения модель сохраняется в файл *ppo_adroit_hammer*.

В листинге 3.3 представлен код, который выполняет тестирование модели с подкреплением.

Листинг 3.3 — Тестирование

```
obs, _ = env.reset()
done, truncated = False, False
while not (done or truncated):
    action, _ = model.predict(obs, deterministic=True)
    obs, reward, done, truncated, _ = env.step(action)
    print(f"REWARD: {reward}")
obs, _ = env.reset()
```

Окружение сбрасывается в начальное состояние. Начинается цикл взаимодействия модели с окружением, который продолжается до тех пор, пока эпизод не завершится или не будет прерван. Модель на основе текущего наблюдения предсказывает действие. Выбранное действие применяется в среде. Метод *step* возвращает новое наблюдение, награду, флаги завершения и прерывания. Значение полученной награды выводится на печать. Если эпизод завершился или был прерван, среда сбрасывается, и цикл прерывается.

3.3 Тестирование и результаты

В ходе обучения модель выполнила 1 000 000 шагов. На графике 3.1 отображен процесс получения наград агентом в ходе обучения. На оси абсцисс отложено количествово шагов обучения, а по оси ординат полученная суммарная награда к данному количеству шагов. На графике прослеживается успешное обучение модели PPO в среде AdroitHandHammer-v1. Начав с периода исследования и низких наград, агент постепенно вырабатывает эффективную стратегию, что отражается в значительном и продолжительном росте суммарной награды на протяжении большей части процесса обучения. Это свидетельствует о том, что модель научилась успешно управлять рукой-роботом для выполнения задачи забивания гвоздя.

Модель была протестирована на 500 итерациях. На рисунке 3.2 показана визуализация успешного выполнения работы агента в среде AdroitHandHammer-v1.

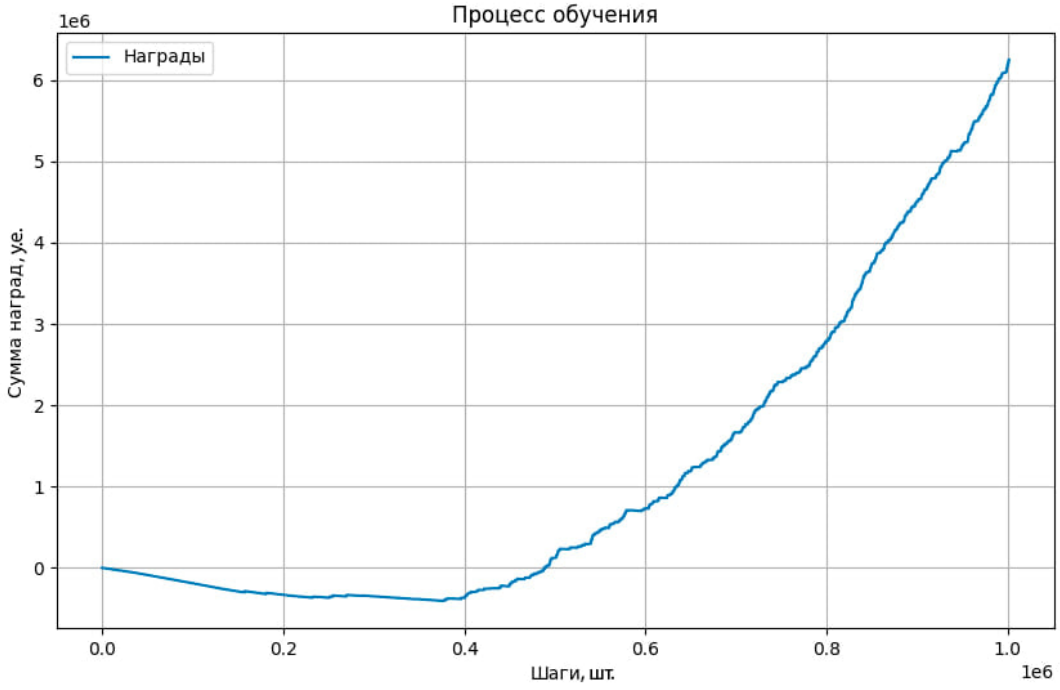
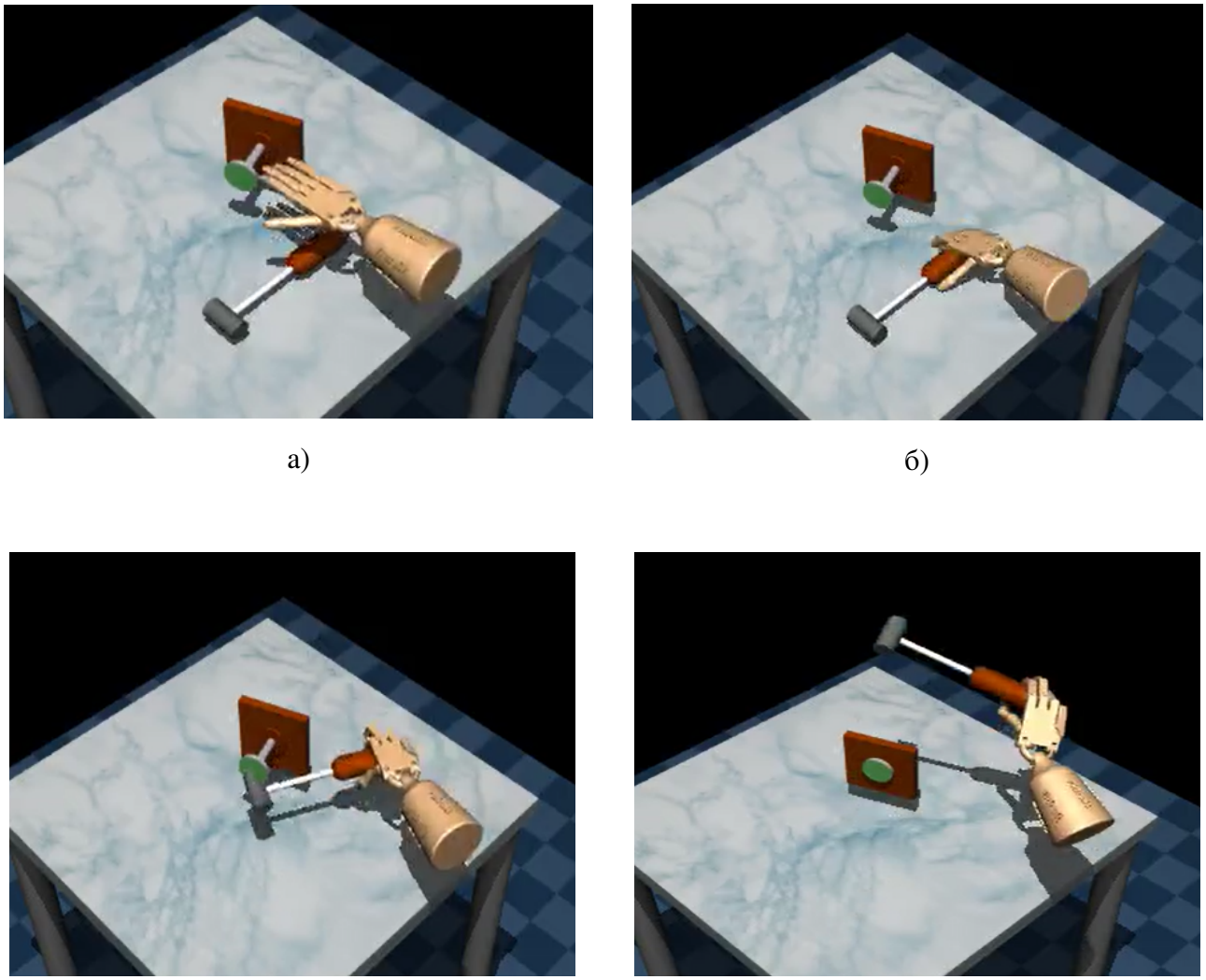


Рисунок 3.1 — Полученные награды в процессе обучения



а) б)

в) г)

Рисунок 3.2 — Визуализация работы агента: а) начало; б) поднятие молотка; в) молоток возле гвоздя; г) конец

Вывод

В ходе технологической части модель была успешно обучена для решения задачи AdroitHand - Hammer-v1. Эта модель может быть использована для управления рукой-роботом, чтобы выполнить задачу забивания гвоздя. Анализ процесса обучения модели показал, что модель успешно вырабатывает эффективную стратегию для выполнения задачи, а тестирование доказало на практике, что агент успешно применяет полученную стратегию для выполнения задачи.

ЗАКЛЮЧЕНИЕ

В рамках данной лабораторной работы была успешно решена задача разработки алгоритма управления роботизированной кистью Adroit для выполнения задачи Adroit Hammer с использованием методов обучения с подкреплением.

В аналитическом разделе был обоснован выбор алгоритма PPO, показавшего оптимальное сочетание стабильности и эффективности для поставленной задачи.

В конструкторском разделе представлена разработанная модель обучения агента, включая этапы инициализации, тренировки, тестирования и визуализации, а также описание среды AdroitHandHammer-v1.

В технологическом разделе описана программная реализация алгоритма, включающая инициализацию среды и агента, настройку обратного вызова и процесс обучения. Результаты обучения демонстрируют стабильный рост суммарной награды, что свидетельствует об эффективности выбранного алгоритма и успешном освоении агентом необходимых навыков. Визуализация действий агента подтверждает, что агент научился манипулировать молотком и достигать целевой позиции гвоздя.

Таким образом, цель лабораторной работы достигнута – разработан алгоритм управления роботизированной кистью Adroit, позволяющий успешно решать задачу Adroit Hammer с использованием методов обучения с подкреплением.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Саттон Р. С. Обучение с подкреплением / Р. С. Саттон, Э. Г. Барто; пер. с англ. – Москва : Бином. Лаборатория знаний, 2017. – 400 с.

2. Кэблинг Л. П. Обучение с подкреплением: обзор / Л. П. Кэблинг, М. Л. Литтман, Э. У. Мур // Journal of artificial intelligence research. – 1996. – Т. 4. – С. 237–285.

3. *Gymnasium Robotics Documentation* [Электронный ресурс]. – Режим доступа: https://robotics.farama.org/ (дата обращения 26.01.2025)

4. *Pycharm 2023 Community edition - среда разработки* [Электронный ресурс]. – Режим доступа: https://www.jetbrains.com/ru-ru/pycharm/download/other.html (дата обращения 26.01.2025)

5. *Python 3.12 Documentation* [Электронный ресурс]. – Режим доступа: https://docs.python.org/3/whatsnew/3.12.html (дата обращения 26.01.2025)

6. *Gymnasium Documentation* [Электронный ресурс]. – Режим доступа: https://gymnasium.farama.org/ (дата обращения 26.01.2025)

7. *Stable-Baselines3 Documentation* [Электронный ресурс]. – Режим доступа: https://stable-baselines3.readthedocs.io/en/master/ (дата обращения 26.01.2025)

8. *NumPy Documentation* [Электронный ресурс]. – Режим доступа: https://numpy.org/doc/stable/user/quickstart.html (дата обращения 26.01.2025)

9. *Matplotlib 3.9.2 documentation* [Электронный ресурс]. – Режим доступа: https://matplotlib.org/stable/index.html (дата обращения 26.01.2025)

10. *Pathlib Documentation* [Электронный ресурс]. – Режим доступа: https://docs.python.org/3/library/pathlib.html (дата обращения 26.01.2025)