ВЕСТНИК НАЦИОНАЛЬНОГО ИССЛЕДОВАТЕЛЬСКОГО ЯДЕРНОГО УНИВЕРСИТЕТА “МИФИ”, 2022, том 11, № 2, с. 143–152

ПРИКЛАДНАЯ МАТЕМАТИКА И ИНФОРМАТИКА

УДК 004.896

НЕЙРОСЕТЕВАЯ МОДЕЛЬ УПРАВЛЕНИЯ ДВИЖЕНИЕМ АГЕНТА В ЗАДАЧЕ “СЛЕДОВАНИЕ ЗА ЛИДЕРОМ”, ПОСТРОЕННАЯ НА ПРИНЦИПАХ ОБУЧЕНИЯ С ПОДКРЕПЛЕНИЕМ

© 2022 г. А. В. Грязнов[1], А. А. Селиванов[1], Р. Б. Рыбка[1], В. А. Шеин[1], М. С. Скороходов[1], А. Г. Сбоев[1,2,*]

[1] НИЦ “Курчатовский институт”, Москва, 123182 Россия

[2] Национальный исследовательский ядерный университет “МИФИ”, Москва, 115409 Россия

**e-mail: sag111@mail.ru*

Поступила в редакцию 16.06.2022 г.
После доработки 22.06.2022 г.
Принята к публикации 28.06.2022 г.

В работе исследуется решение малоизученной задачи следования за лидером с помощью нейросетевой модели, настройка которой выполнена на основе обучения с подкреплением (RL). В качестве алгоритма обучения с подкреплением нейросетевой модели (RL-модели) выбран алгоритм proximal policy optimization. Для реализации обучения был разработан эмулятор среды решения задачи. Эмулятор позволяет настраивать среду с различным количеством препятствий, маршрутами различной длины и сложности, а также конфигурировать желаемое поведение агента-ведомого, следующего за лидером. Ввиду быстродействия разработанного эмулятора он, в частности, дает возможность тренировки RL-моделей следования за лидером, процесс настройки которых требует большого числа итераций обучения для различных по сложности маршрутов и препятствий в среде. Представлены результаты сравнительного исследования по выбору признаков, характеризующих текущее наблюдаемое окружение агента, для RL-модели. Показано, что модель, обученная по принципам обучения с подкреплением на наборе признаков лучевых сенсоров, позволяет существенно повысить точность решения задачи, достигая 77% успешного выполнения маршрутов, ошибаясь в большинстве случаев при движении лидера в обратную сторону.

Ключевые слова: обучение с подкреплением, имитационное моделирование, управление движением, нейронные сети

DOI: 10.56304/S2304487X22020055

1. ВВЕДЕНИЕ

Управление робототехническими устройствами в осложненных условиях требует решения комплекса поведенческих задач, частным случаем которых является задача следования агента за ведущим-лидером. Сложность задачи в том, что лидер в процессе движения динамически генерирует маршрут для агента следования, задача которого держаться на заданном расстоянии. Данная задача является актуальной при автоматизации следования в колонне или перемещений в опасных регионах (лед, горы, болота и т.д.).

Существующие алгоритмы, такие как TEB, DWA, EBAND, пригодные для реализации следования по заранее заданному маршруту в полной мере не адаптированы для решения выбранной задачи. В частности, требуются улучшения по ряду позиций: минимизации отклонения от траектории (TEB), учета динамических препятствий (DWA, EBAND), работе с большим количеством препятствий и опасных участках местности за ограниченное время (все алгоритмы). В этой связи перспективным является применение методов обучения с подкреплением, обучаемых желаемому поведению непосредственно с учетом сложностей среды и заданных условий задачи. Целью данной работы является решение задачи следования за лидером на основе модели управления агентом, обученной на принципах обучения с подкреплением. Создание модели следования агента за движущимся ведущим объектом на базе технологии обучения с подкреплением требует выполнения большого количества итераций, включающих обработку наблюдения (St) и генерацию действия (At) непосредственно в среде функционирования агента. Поэтому в данной работе в качестве среды для настройки модели используется 2D-модель мира, эмулирующая необходимые условия решения задачи (см. раздел 3). Такой подход является базовым для обучения

Таблица 1. Результаты тестирования моделей, обученных с разными настройками среды и сенсоров

Среда	Конфигурация признаков	Количество пройденных маршрутов из теста
А	v1	42
	v2	36
	v3	62
Б	v1	35
	v2	41
	v3	77
В	v1	24
	v2	30
	v3	35

RL-моделей, что подтверждается представленным в разделе 2 анализом современных исследований. В качестве алгоритма настройки RL-модели выбран proximal policy optimization (PPO), демонстрирующий эффективность для решения ряда поведенческих задач [1–7]. Описание модели представлено в разделе 4. В разделе 5 представлены результаты сравнительного исследования эффективности различных конфигураций сетей и признаков при решении задачи “следования за лидером”. В целом предлагаемая к исследованию постановка расширяет прикладные задачи анализа данных для робототехнических комплексов [8, 9].

2. ПРЕДЫДУЩИЕ РАБОТЫ

В задаче следования за лидером, как подклассе задач ориентирования робота в пространстве, важным является определение сенсорной системы робота, которая может базироваться как на отдельном виде сенсоров, так и на их совокупности, в частности популярными решениями являются камеры, 3D-камеры, сонары, лидары, датчики температуры и тепловизоры, инфракрасные датчики, гиростабилизаторы, GPS-модули и так далее. Эффективным подходом к определению конфигурации оборудования и особенностей решения задачи является имитационное моделирование [10]: наличие математической модели, которая достаточно точно описывает задачу, открывает возможности для использования настраиваемых алгоритмов, в частности нейронных сетей. Такие модели адаптируются к задаче [11], при обобщении информации становятся способны принимать верные решения в ситуациях, с которыми ранее не сталкивались, но требуют проведения большого числа симуляций [12]. При наличии математической модели такие симуляции могут быть проведены в вычислительной среде с последующей возможной донастройкой алгоритма уже на задаче в физическом мире. Использование данного подхода в робототехнике позволило эффективно решить ряд задач: манипуляции над объектами роботизированной рукой [13, 14], движение четырехногого робота по различным типам местности [15], прыжки четырехногого робота [16], слепой подъем по лестнице двуногого робота [17], гонки дронов по сложным трассам [18] и другие [19–23].

В научной литературе представлен ряд работ, в которых исследуется задача следования в постановке обучения с подкреплением [24–26]. Однако ни в одной из них не представлен исходный код среды моделирования для обучения RL-модели и коррекции постановки задачи.

В качестве альтернативы могут быть использованы программные средства с открытым исходным кодом, которые реализует общепринятый интерфейс для обучения с подкреплением – Gym [27]. Среди таких решений можно выделить: WeBots [28], pybullet robo envs [29], gym-ignition [30], Gazebo [31] и OpenAI ROS [32]. Эти среды реализуют сложную физику и позволяют имплементировать различные ситуации. Однако, для обучения с подкреплением целесообразным является создание более простой среды, которая моделирует непрерывное двумерное пространство, для формализации желаемого поведения при решении задачи следования и определения функции расчета награды. Полученные в результате симуляции модели и формализации поведения могут быть использованы для последующего переноса и дообучения в более сложной среде, более детально моделирующей кинематику и динамику твердого тела.

3. ОПИСАНИЕ СРЕДЫ

Среда представляет собой двумерное евклидово пространство (2D-мир) со статичными и подвижными объектами (см. рис. 1). К статичным объектам относятся препятствия двух типов: случайно расположенные препятствия заданного размера и препятствия, имитирующие узкие места (например, мосты/броды/горные перевалы и т.д.). Подвижными объектами являются ведущий и ведомый. Управление подвижными объектами осуществляется установкой значений линейной скорости и скорости поворота для каждого из них в каждый момент времени t. Скорости ведущего устанавливаются на основе заданного ведущему маршрута следования и выбранных диапазонов ускорений и расписания скоростей. Скорости ведомого должны быть получены из модели следования, обучаемой непосредственно в среде желаемому поведению.

Желаемое поведение ведомого должно удовлетворять нескольким условиям. Ему необходимо: а) двигаться по маршруту, который проходит ве-

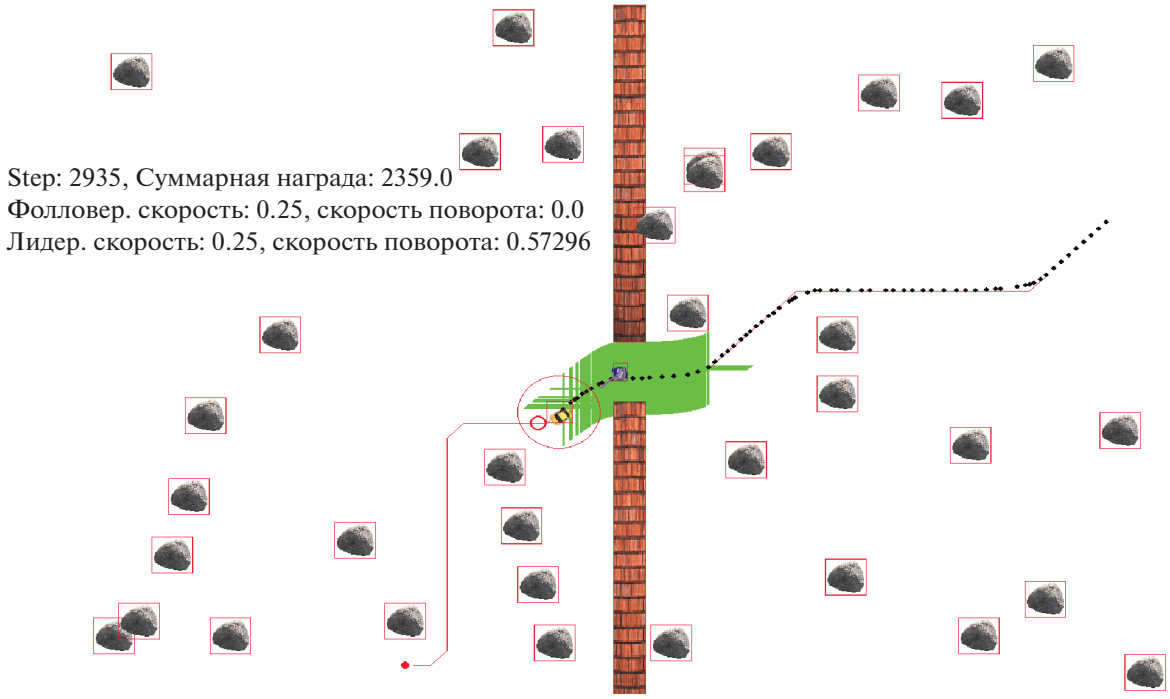

Рис. 1. Визуализация среды. Желтая машина – лидер, синяя машина – ведомый, изображения камней и стены – препятствия, красные прямоугольники вокруг объектов – границы для определения столкновений, красная точка в конце пути – целевая позиция лидера, красная линия – планируемый путь лидера, черные точки – фактический путь лидера, красная окружность вокруг лидера – минимальное расстояние, на котором надо держаться ведомому, красная окружность перед лидером – следующая промежуточная позиция лидера, зеленая зона – зона допустимого отклонения от маршрута и допустимого расстояния до лидера.

дущий, с минимальным отклонением; б) находиться на определенном расстоянии, не приближаясь и не удаляясь дальше заданных границ; в) регулировать скорость и направление движения в зависимости от изменения скорости и направления ведущего объекта; г) избегать столкновений с ведущим и прочими объектами.

Для оценки поведения ведомого, в каждый момент времени t, вычисляется награда на основе следующих показателей:

– d_{min} – дистанция, ближе которой ведомый не должен приближаться к ведущему;

– d_{max} – дистанция, дальше которой ведомый не должен удаляться от ведущего;

– dev – максимальное расстояние от ведомого до ближайшей точки маршрута ведущего;

– d_{route} – текущее расстояние от ведомого до ближайшей точки маршрута;

– d_{leader} – расстояние от ведомого до ведущего, вычисляемое как сумма расстояний между точками маршрута от ведущего до ближайшей к ведомому точке маршрута;

– ε_{pos} – расстояние, на котором объекты считаются находящимися в одной и той же точке.

В качестве базовой функции расчета награды реализована следующая:

– если $d_{\text{leader}} <= d_{\text{min}}$, значит ведомый находится слишком близко: $reward_t = reward_t - 5$;

– иначе если $d_{\text{route}} <= \varepsilon_{pos}$, значит ведомый находится на маршруте:

  ○ если $d_{\text{min}} <= d_{\text{leader}} <= d_{\text{max}}$, значит ведомый держит нужную дистанцию от лидера на маршруте: $reward_t = reward_t + 1$;

  ○ иначе если $d_{\text{leader}} > d_{\text{max}}$, значит ведомый отстал от ведущего: $reward_t = reward_t + 0.1$;

– иначе если $d_{\text{leader}} > d_{\text{max}}$, значит ведомый отстал от ведущего: $reward_t = reward_t - 1$;

– иначе если $d_{\text{route}} <= dev$, значит ведомый находится в пределах допустимого расстояния от маршрута ведущего, но не на самом маршруте:

  ○ если $d_{\text{min}} <= d_{\text{leader}} <= d_{\text{max}}$, значит ведомый держит нужную дистанцию от лидера на маршруте: $reward_t = reward_t + 0.5$;

– иначе если $d_{\text{route}} > dev$, значит ведомый сильно отклонился от маршрута $reward_t = reward_t - 1$;

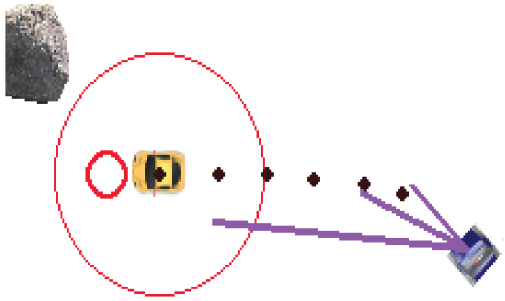

Рис. 2. Визуализация работы радара истории маршрута. Фиолетовые линии проведены через центр сектора на длину, равную расстоянию до ближайшей точки, попавшей в этот сектор.

Рис. 3. Визуализация работы лидара. Красными точками отмечены крайние точки по каждому из направлений, которые не закрыты препятствием.

– если границы объекта, который обозначает ведомого, пересеклись с границами объекта-ведущего или объекта-препятствия, $reward_t = reward_t - 10$ и симуляция мгновенно заканчивается (считается, что произошла авария).

Время в среде измеряется “кадрами” (frame). В среде реализован вариант, когда действие объекта длится в течение одного “шага” (step), которому соответствует переменное число “кадров”, что эмулирует возможные задержки в передаче данных. Модель, которая определяет поведение агента, получает состояние среды (наблюдение) раз в шаг. Действие агента выбирается один раз за шаг и реализуется каждый кадр в течение этого шага. В эмулируемой среде задан масштаб расстояния 1 метр = 50 пикселей.

Полная конфигурация среды приведена на сайте проекта [38].

Одна симуляция (“simulation”, “episode”) включает в себя конечную последовательность шагов, которая завершается при удовлетворении одного из следующих критериев:

– ведущий успешно завершил маршрут;

– ведущий попал в аварию;

– ведомый попал в аварию;

– число шагов симуляции превысило допустимый лимит (по умолчанию: 10000 шагов);

– расстояние между ведомым и ведущим превысило $d_{\max}k$ (по умолчанию $k = 1.4$);

– в течение нескольких последних шагов не было получено положительной награды и был получен штраф на сумму –300.

Наблюдение с каждого шага включает информацию с датчиков ведомого и получаемую из среды информацию о ведущем. В среде ведомому доступны следующие варианты сенсоров:

1) Радар, реагирующий на точки из истории маршрута ведущего. История передвижений лидера обновляется на каждом восьмом шаге симуляции. Максимально может храниться не более 5000 точек истории. Т.к. история маршрута может включать различное количество точек в разные моменты времени, проводится предобработка. Сектор 180 градусов перед ведомым делится на N секторов (по умолчанию $N = 9$). На каждом шаге для каждого сектора выбирается ближайшая позиция из истории, которая в него попала. Расстояние до ближайших точек каждого сектора возвращается как результат наблюдений сенсора. Если нет точек, попавших в сектор, для него возвращается 0. Перед подачей в модель, значения делятся на $d_{\max}$ и ограничиваются отрезком от 0 до 1. Пример признаков радара истории представлен на рис. 2.

2) Лидар. Область перед ведомым с заданным углом обзора 100 градусов разделяется на границы с шагом 5 градусов, каждая граница разделяется на 30 точек. Лидар позволяет получить информацию от каждой точки не закрытой препятствием или точки, максимально удаленной от ведомого, и не закрытой от него препятствием. Пример признаков представлен на рис. 3.

4. ОПИСАНИЕ АЛГОРИТМА ОБУЧЕНИЯ

Proximal policy optimization (PPO) [33] – это алгоритм обучения с подкреплением, который а) не моделирует полностью окружающий мир и его динамику (model-free) и б) проводит обновление весов только опираясь на историю переходов с использованием последнего варианта модели (on-policy), в) использует архитектуру актор-критик, включающую модель поведения (согласно которой на основе наблюдений выбирается действие агента) и модель-критик (предсказывающую ожидаемую награду на основе наблюдений). Метод является более стабильным и эффективным в плане использования накопленных данных

чем стандартный policy gradient алгоритм [34] и при этом имеет более простую реализацию, чем алгоритм trust region policy optimization [35], от которого была унаследована оптимизируемая целевая функция (см. формулу 1). В ходе обучения веса модели модифицируются для максимизации следующей целевой функции:

$$L^{CLIP}(\theta) = \mathbb{E}_t[\min(r_t(\theta)\times \\ \times A_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\times A_t)] \quad (1)$$

Здесь $r_t(\theta) = \pi_\theta(a_t|s_t)/\pi_{\theta_{\text{old}}}(a_t|s_t)$ – отношение между логарифмами вероятностей выбранных действий a_t для наблюдений s_t для новой π_θ с обновленными весами и предыдущей $\pi_{\theta_{old}}$, которая была использована при сборе истории; θ – веса policy модели, A_t – значения преимущества (advantage), рассчитанные алгоритмом generalized advantage estimation (GAE) [36] и предсказанными значениями модели-критика, ϵ – гиперпараметр, используемый для ограничения целевой функции, позволяющий избегать слишком больших изменений весов модели.

Итерация обучения в общем случае выглядит следующим образом:

1. Сбор батча данных, в ходе которого агент исследует среду, выбирая действия с использованием текущей модели поведения $\pi_{\theta_{\text{old}}}$.

2. Расчет значения advantage с использованием текущей модели-критика V_ϕ.

3. Проведение итерации стохастического градиентного спуска для обновления весов модели поведения π_θ максимизируя целевую функцию L^{CLIP}.

4. Обновление веса модели-критика V_ϕ минимизируя среднеквадратическую ошибку предсказаний.

Нами использовалась реализация алгоритма ppo из библиотеки ray-rllib [37]. Настройки алгоритма обучения приведены на сайте проекта [38].

5. ОПИСАНИЕ НЕЙРОСЕТЕВЫХ МОДЕЛЕЙ

Входом для используемых моделей является вектор наблюдений, полученный от сенсоров робота. Перед подачей значения сенсоров нормируются в диапазон от 0 до 1. В качестве входных данных рассмотрено несколько вариантов признаков, сформированных из доступных среды:

1) признаки радара;

2) признаки радара и лидара;

3) признаки радара и лучевой сенсор.

На выходе модель должна предсказать два значения: линейную скорость и скорость поворота.

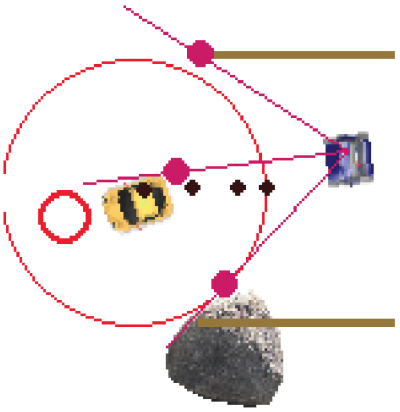

Рис. 4. Визуализация работы лучевого сенсора. Красные лучи – лучи сенсоров. Красные точки – пересечения лучей с границами зоны безопасного отклонения и препятствий.

Для работы используются две нейросетевых модели: 1) модель поведения и 2) модель-критик, которые состоят из трех слоев. Первые 2 слоя, обрабатывающие входные наблюдения имеют по 256 нейронов с функцией активации в виде гиперболического тангенса. Выходной слой модели-критика содержит 1 нейрон с линейной активацией. Выходной слой модели поведения содержит 4 нейрона с линейной активацией, по 2 выхода для каждого действия (угловая и линейная скорости). Эти два выхода соответствуют среднему значению скорости и логарифму стандартного отклонения, задавая распределение вероятностей, из которого выбирается действие на этапе сбора данных для обучения. При тестировании модели используются только значения первых выходов для каждого действия.

6. ЭКСПЕРИМЕНТЫ

В качестве моделей рассматривались три варианта с наборами параметров (см. раздел “Описание моделей”). Модели обучались в эмуляционных средах в различных конфигурациях:

А) с числом препятствий = 35, маршрут ведущего строился от начальной точки на правой половине мира в конечную точку на левой половине мира (см. рис. 5);

Б) с числом препятствий = 70, маршрут ведущего строился от начальной точки на правой половине мира в конечную точку на левой половине мира (см. рис. 6);

В) с числом препятствий = 35, маршрут ведущего начинал строится от начальной точки на правой половине мира, в следующую точку на левой половине мира, а затем еще в две точки, которые могут быть на обеих половинах, но каждая из

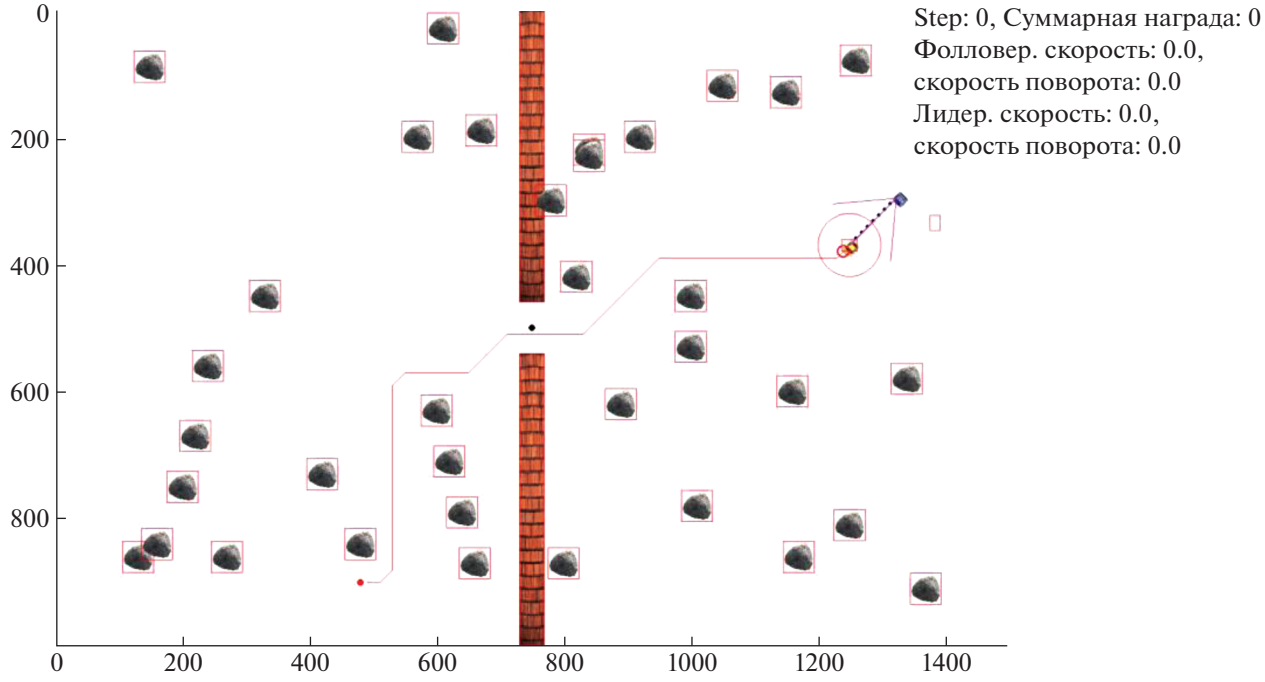

Рис. 5. Эмуляционный мир с числом препятствий = 35, маршрут ведущего строился из правой части мира в левую.

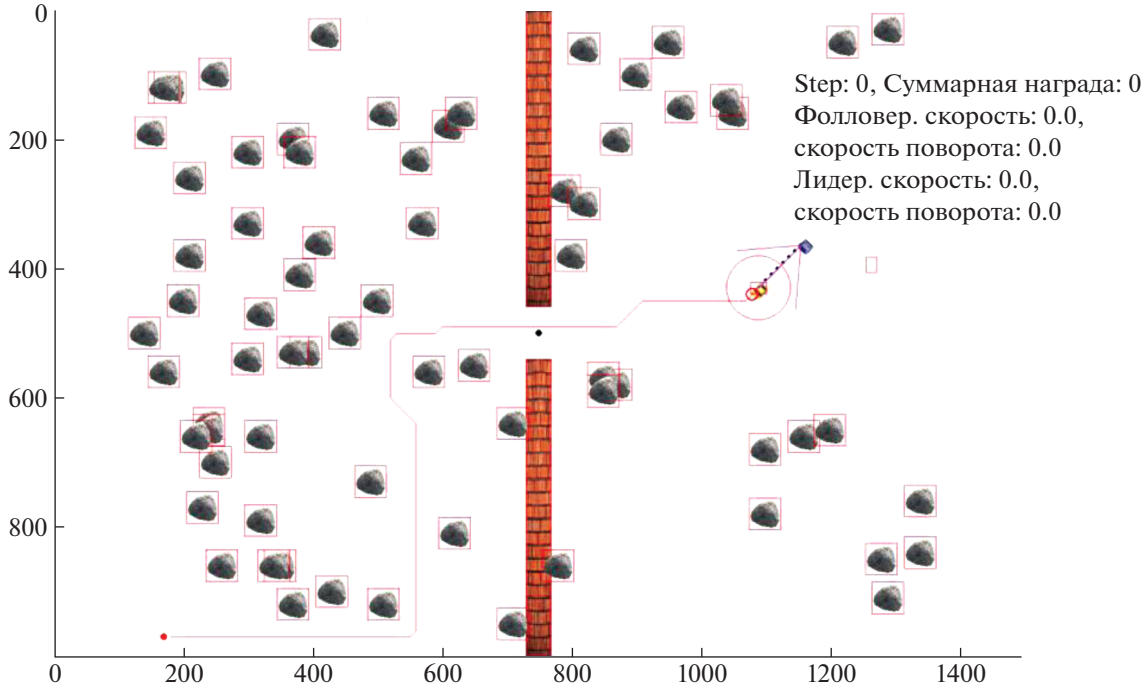

Рис. 6. Эмуляционный мир с числом препятствий = 70, маршрут ведущего строился из правой части в левую.

них генерируется на противоположной половине по вертикали от предыдущей (см. рис. 7).

Для оценки моделей было зафиксировано 100 маршрутов при заданном числе препятствий (35), расположенных случайно. Критерием эффективности было количество пройденных тестовых маршрутов.

Модели обучались в течение 800 итераций. На каждой итерации собиралась выборка из 6000 шагов, что соответствует примерно 10–20 эпизодам.

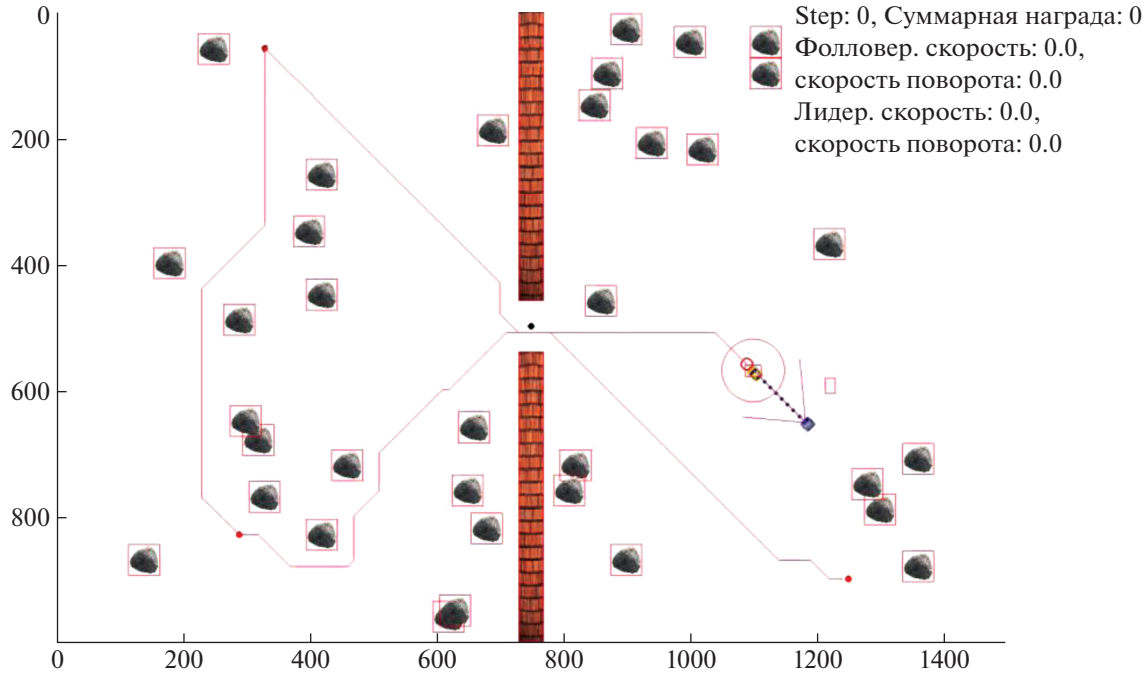

Рис. 7. Эмуляционный мир с числом препятствий = 35, маршрут ведущего строился через 3 точки.

После чего проводилось 30 итераций обучения на полученной выборке. Время обучения модели занимало около 30 часов с учетом использования четырех процессов с копиями среды для сбора обучающего батча данных, запущенных на четырех ядрах процессора Intel(R) Xeon(R) CPU E5-2650 v2 @ 2.60GHz, с использованием одного графического процессора Nvidia Tesla K80 (12GB) для обучения.

Таким образом, лучшие результаты достигаются при обучении модели в среде “Б” при использовании конфигурации признаков № 3.

7. ВЫВОДЫ

В работе представлены результаты исследования по созданию нейросетевой модели управления движением агента в среде для решения задачи следования за лидером. Создание модели основано на алгоритме обучения с подкреплением, применение которого возможно за счет разработанной среды эмуляции 2D мира и базовых условий для решения задачи. Показано, что наибольшую эффективность достигает модель с признаками лучевых сенсоров до границ области безопасного отклонения от эталонного маршрута лидера и\или препятствий. Разработанный подход показал эффективность при 77% тестовых маршрутах, что может быть основой для его использования в реальных условиях.

БЛАГОДАРНОСТИ

Работа выполнена при поддержке НИЦ “Курчатовский институт” (приказ № 2754 от 28.10.2021) с использованием вычислительных ресурсов ОВК НИЦ “Курчатовский институт”, http://computing.nrcki.ru/.

СПИСОК ЛИТЕРАТУРЫ

1. *Raffin A., Kober J., Stulp F.* Smooth exploration for robotic reinforcement learning //Conference on Robot Learning. PMLR, 2022. P. 1634–1644.
2. *Open AI, Berner C., Brockman G., Chan B., Cheung V., Dębiak P., Dennison C., Farhi D., Fischer Q., Hashme S., Hesse C., Józefowicz R., Gray S., Olsson C., Pachocki J., Petrov M., Pinto H. P. d. O., Raiman J., Salimans T., Schlatter J., Schneider J., Sidor S., Sutskever I., Tang J., Wolski F., Zhang S.* Dota 2 with large scale deep reinforcement learning //arXiv preprint arXiv:1912.06680. 2019.
3. *You J., Liu B., Ying R., Pande V., Leskovec J.* Graph convolutional policy network for goal-directed molecular graph generation // Advances in neural information processing systems, 2018.
4. *Juliani A., Berges V.-P., Teng E., Cohen A., Harper J., Elion C., Goy C., Gao Y., Henry H., Mattar M., Lange D.* Unity: A general platform for intelligent agents // arXiv preprint arXiv:1809.02627, 2018.
5. *Tan J., Zhang T., Coumans E., Iscen A., Bai Y., Hafner D., Bohez S., Vanhoucke V.* Sim-to-real: Learning agile locomotion for quadruped robots //arXiv preprint arXiv:1804.10332, 2018.

6. *Burda Y., Edwards H., Pathak D., Storkey A., Darrell T., Efros A.A.* Large-scale study of curiosity-driven learning //arXiv preprint arXiv:1808.04355, 2018
7. *Kaiser L., Babaeizadeh M., Milos P., Osinski B., Campbell R.H., Czechowski K., Erhan D., Finn C., Kozakowski P., Levine S., Mohiuddin A., Sepassi R., Tucker G., Michalewski H.* Model-based reinforcement learning for atari //arXiv preprint arXiv:1903.00374, 2019.
8. *Молошников И.А., Грязнов А.В., Власов Д.С., Рыбка Р.Б., Сбоев А.Г.* Применение мультизадачной модели для практических задач генерации заголовка, определения лемм и ключевых слов // Вестник НИЯУ, 2020. Т. 9. № 3. С. 236–244.
9. *Сбоев А.Г., Давыдов Ю.А., Рыбка Р.Б.* Нейросетевая модель для перевода текстовых команд мобильному роботу на естественном русском языке в семиотический формат RDF // Лазерные, плазменные исследования и технологии-ЛАПЛАЗ-2021, 2021. С. 138–139.
10. *Muratore F., Ramos F., Turk G., Yu W., Gienger M., Peters J.* Robot learning from randomized simulations: A review //arXiv preprint arXiv:2111.00956, 2021.
11. *Filos A., Tigas P., McAllister R., Rhinehart N., Levine S., Gal Y.* Can autonomous vehicles identify, recover from, and adapt to distribution shifts? // International Conference on Machine Learning. PMLR, 2020.
12. *Silver D., Hubert T., Schrittwieser J., Antonoglou I., Lai M., Guez A., Hassabis D.* A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play //Science, 2018. V. 362. Is. 6419. P. 1140–1144.
13. *Akkaya I., Andrychowicz M., Chociej M., Litwin M., McGrew B., Petron A., Paino A., Plappert M., Powell G., Ribas R., Schneider J., Tezak N., Tworek J., Welinder P., Weng L., Yuan Q., Zaremba W., Zhang L.* Solving rubik's cube with a robot hand //arXiv preprint arXiv:1910.07113, 2019.
14. *Saeed M., Nagdi M., Rosman B.* Deep reinforcement learning for robotic hand manipulation // 2020 International Conference on Computer, Control, Electrical, and Electronics Engineering (ICCCEEE). IEEE, 2021.
15. *Joonho L., Jemin H., Lorenz W., Vladlen K., Marco H.* Learning quadrupedal locomotion over challenging terrain // Science robotics 5.47, 2020: eabc5986.
16. *Bellegarda G., Quan N.* Robust quadruped jumping via deep reinforcement learning //arXiv preprint arXiv:2011.07089, 2020.
17. *Siekmann J., Green K., Warila J., Fern A., Hurst J.* Blind bipedal stair traversal via sim-to-real reinforcement learning //arXiv preprint arXiv:2105.08328, 2021.
18. *Song Y., Steinweg M., Kaufmann E., Scaramuzza D.* Autonomous drone racing with deep reinforcement learning // 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2021.
19. *Azar A.T., Koubaa A., Ali M.N., Ibrahim H.A., Ibrahim Z.F., Kazim M., Ammar A., Benjdira B., Khamis A.M., Hameed I.A., Casalino G.* Drone deep reinforcement learning: A review //Electronics 10.9, 2021. P. 999.
20. *Longfei Y., Rennong Y., Ying Z., Lixin Y., Zhuangzhuang W.* Deep Reinforcement Learning for UAV Intelligent Mission Planning // Complexity 2022. V. 2022. doi: 10.1155/2022/3551508.
21. *Liu C., Erik-Jan V.K.* HER-PDQN: A Reinforcement Learning Approach for UAV Navigation with Hybrid Action Spaces and Sparse Rewards // AIAA SCITECH 2022 Forum, 2022.
22. *Salvato E., Fenu G., Medvet E., Pellegrino F.A.* Crossing the Reality Gap: a Survey on Sim-to-Real Transferability of Robot Controllers in Reinforcement Learning // IEEE Access, 2021.
23. *Xie J., Zhou R., Liu Y., Luo J., Xie S., Peng Y., Pu H.* Reinforcement-Learning-Based Asynchronous Formation Control Scheme for Multiple Unmanned Surface Vehicles // Applied Sciences 11.2, 2021. P. 546.
24. *Zhou Y., Lu F., Pu G., Ma X., Sun R., Chen Hsi-Yuan, Li X., Wu D.* Adaptive leader-follower formation control and obstacle avoidance via deep reinforcement learning // 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2019.
25. *Md Suruz Miah, Amr Elhussein, Keshtkar F., Abouheaf M.* Model-free reinforcement learning approach for leader-follower formation using nonholonomic mobile robots // The Thirty-Third International Flairs Conference, 2020.
26. *Deka A., Luo W., Li H., Lewis M., Sycara K.* Hiding Leader's Identity in Leader-Follower Navigation through Multi-Agent Reinforcement Learning // 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2021.
27. *Brockman G., Cheung V., Pettersson L., Schneider J., Schulman J., Tang J., Zaremba W.* Openai gym // arXiv preprint arXiv:1606.01540, 2016.
28. *Michel O.* Cyberbotics ltd. webots™: professional mobile robot simulation // International Journal of Advanced Robotic Systems, 2004. V. 1. Is. 1. P. 5.
29. *Coumans E., Bai Y.* Pybullet, a python module for physics simulation for games, robotics and machine learning, 2016.
30. *Ferigo D., Traversaro S., Metta G., Pucci D.* Gym-ignition: Reproducible robotic simulations for reinforcement learning // 2020 IEEE/SICE International Symposium on System Integration (SII), IEEE, 2020. P. 885–890.
31. *Nathan K., Howard A.* Design and use paradigms for gazebo, an open-source multi-robot simulator // 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2004. (IEEE Cat. No. 04CH37566).
32. *Zamora I., Lopez N.G., Vilches V.M., Cordero A.H.* Extending the openai gym for robotics: a toolkit for reinforcement learning using ros and gazebo // arXiv preprint arXiv:1608.05742, 2016.
33. *Schulman J., Wolski F., Dhariwal P., Radford A., Klimov O.* Proximal policy optimization algorithms // arXiv preprint arXiv:1707.06347v2, 2017.
34. *Sutton R.S., McAllester D., Singh S., Mansour Y.* Policy gradient methods for reinforcement learning with function approximation // Advances in neural information processing systems, 1999. V. 12.
35. *Schulman J., Levine S., Moritz P., Jordan M.I., Abbeel P.* Trust region policy optimization // CoRR, abs/1502.05477, 2015.

36. *Schulman J., Moritz P., Levine S., Jordan M., Abbeel P.* High-dimensional continuous control using generalized advantage estimation // arXiv preprint arXiv:1506.02438, 2015.

37. *Liang E., Liaw R., Nishihara R., Moritz P., Fox R., Goldberg K., Stoica I.* RLlib: Abstractions for distributed reinforcement learning. In International Conference on Machine Learning, 2018. P. 3053–3062.

38. A continuous environment for reinforcement learning of the task of the following the leader. [Электронный ресурс]. https://github.com/sag111/ContiniousEnvironment_Follower_Leader (дата обращения 10.06.2022)

Vestnik Natsional'nogo Issledovatel'skogo Yadernogo Universiteta "MIFI", 2022, vol. 11, no. 2, pp. 143–152

Reinforcement Learning Neural Network Model for Agent Motion Control in the Leader–Follower Problem

A. V. Gryaznov[a], A. A. Selivanov[a], R. B. Rybka[a], V. A. Shein[a], M. S. Skorokhodov[a], and A. G. Sboev[a,b,#]

[a] *National Research Center Kurchatov Institute, Moscow, 123182 Russia*

[b] *National Research Nuclear University MEPhI (Moscow Engineering Physics Institute), Moscow, 115409 Russia*

[#] *e-mail: sag111@mail.ru*

Received June 16, 2022; revised June 22, 2022; accepted June 28, 2022

Abstract—The solution of the insufficiently studied leader–follower problem has been studied within a reinforcement learning neural network model. The reinforcement learning algorithm chosen for training the neural network model is the proximal policy optimization algorithm. In order to implement the training, an emulator of the environment for the leader–follower problem has been developed. The emulator allows one to set up an environment with a different number of obstacles and routes of different lengths and complexity, as well as to configure the desired behavior of the follower agent following the leader. The developed emulator is performant enough for the feasibility of training the reinforcement learning leader–follower models, the adjustment of which requires a large number of training iterations for routes and obstacles of various complexity in the environment. The presented results include the choice of features characterizing the current observable environment of the agent for the reinforcement learning model. A model trained according to the reinforcement learning principles on a set of features of ray-type range sensors can significantly improve the accuracy of solving the problem, reaching 77% of the successful execution of routes, making mistakes mostly when the leader moves in the opposite direction.

Keywords: reinforcement learning, simulation, motion control, neural networks

DOI: 10.56304/S2304487X22020055

REFERENCES

1. Raffin A., Kober J., Stulp F. Smooth exploration for robotic reinforcement learning. *Conference on Robot Learning. PMLR*, 2022, pp. 1634–1644.
2. Open AI, Berner C., Brockman G., Chan B., Cheung V., Dębiak P., Dennison C., Farhi D., Fischer Q., Hashme S., Hesse C., Józefowicz R., Gray S., Olsson C., Pachocki J., Petrov M., Pinto H. P. d. O., Raiman J., Salimans T., Schlatter J., Schneider J., Sidor S., Sutskever I., Tang J., Wolski F., Zhang S. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680.* 2019.
3. You J., Liu B., Ying R., Pande V., Leskovec J. Graph convolutional policy network for goal-directed molecular graph generation. *Advances in neural information processing systems*, 2018.
4. Juliani A., Berges V.-P., Teng E., Cohen A., Harper J., Elion C., Goy C., Gao Y., Henry H., Mattar M., Lange D. Unity: A general platform for intelligent agents. *arXiv preprint arXiv:1809.02627,* 2018.
5. Tan J., Zhang T., Coumans E., Iscen A., Bai Y., Hafner D., Bohez S., Vanhoucke V. Sim-to-real: Learning agile locomotion for quadruped robots. *arXiv preprint arXiv:1804.10332,* 2018.
6. Burda Y., Edwards H., Pathak D., Storkey A., Darrell T., Efros A.A. Large-scale study of curiosity-driven learning . *arXiv preprint arXiv:1808.04355,* 2018
7. Kaiser L., Babaeizadeh M., Milos P., Osinski B., Campbell R.H., Czechowski K., Erhan D., Finn C., Kozakowski P., Levine S., Mohiuddin A., Sepassi R., Tucker G., Michalewski H. Model-based reinforcement learning for Atari. *arXiv preprint arXiv:1903.00374*, 2019.
8. Moloshnikov I.A., Gryaznov A.V., Vlasov D.S., Rybka R.B., Sboev A.G. Primenenie mul'tizadachnoj modeli dlya prakticheskih zadach generacii zagolovka, opredeleniya lemm i klyuchevyh slov [Application of a

multitasking model for practical problems of title generation, definition of lemmas and keywords]. *Vestnik NIYaU MIFI*, 2020, vol. 9, no. 3, pp. 236–244. (In Russian)
9. Sboev A.G., Davydov Yu.A., Rybka R.B. Nejrosetevaya model' dlya perevoda tekstovyh komand mobil'nomu robotu na estestvennom russkom yazyke v semioticheskij format RDF. [A neural network model for translating text commands to a mobile robot in natural Russian into the semiotic RDF format]. Lazernye, plazmennye issledovaniya i tekhnologii-LAPLAZ-2021 [Laser, plasma research and technology-LAPLAZ-2021], 2021. pp. 138–139. (In Russian)
10. Muratore F., Ramos F., Turk G., Yu W., Gienger M., Peters J. Robot learning from randomized simulations: A review. *arXiv preprint arXiv:2111.00956*, 2021.
11. Filos A., Tigas P., McAllister R., Rhinehart N., Levine S., Gal Y. Can autonomous vehicles identify, recover from, and adapt to distribution shifts? *International Conference on Machine Learning. PMLR,* 2020.
12. Silver D., Hubert T., Schrittwieser J., Antonoglou I., Lai M., Guez A., Hassabis D. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science,* 2018. vol. 362, is. 6419, pp. 1140–1144.
13. Akkaya I., Andrychowicz M., Chociej M., Litwin M., McGrew B., Petron A., Paino A., Plappert M., Powell G., Ribas R., Schneider J., Tezak N., Tworek J., Welinder P., Weng L., Yuan Q., Zaremba W., Zhang L. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.
14. Saeed M., Nagdi M., Rosman B. Deep reinforcement learning for robotic hand manipulation. *2020 International Conference on Computer, Control, Electrical, and Electronics Engineering (ICCCEEE). IEEE*, 2021.
15. Joonho L., Jemin H., Lorenz W., Vladlen K., Marco H. Learning quadrupedal locomotion over challenging terrain. *Science robotics 5.47,* 2020: eabc5986.
16. Bellegarda G., Quan N. Robust quadruped jumping via deep reinforcement learning. *arXiv preprint arXiv:2011.07089*, 2020.
17. Siekmann J., Green K., Warila J., Fern A., Hurst J. Blind bipedal stair traversal via sim-to-real reinforcement learning . *arXiv preprint arXiv:2105.08328*, 2021.
18. Song Y., Steinweg M., Kaufmann E., Scaramuzza D. Autonomous drone racing with deep reinforcement learning. *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE,* 2021.
19. Azar A.T., Koubaa A., Ali M.N., Ibrahim H.A., Ibrahim Z.F., Kazim M., Ammar A., Benjdira B., Khamis A.M., Hameed I.A., Casalino G. Drone deep reinforcement learning: A review. *Electronics 10.9*, 2021, pp. 999.
20. Longfei Y., Rennong Y., Ying Z., Lixin Y., Zhuangzhuang W. Deep Reinforcement Learning for UAV Intelligent Mission Planning. *Complexity 2022*, vol. 2022. doi: 10.1155/2022/3551508.
21. Liu C., Erik-Jan V.K. HER-PDQN: A Reinforcement Learning Approach for UAV Navigation with Hybrid Action Spaces and Sparse Rewards. *AIAA SCITECH 2022 Forum*, 2022.
22. Salvato E., Fenu G., Medvet E., Pellegrino F. A. Crossing the Reality Gap: a Survey on Sim-to-Real Transferability of Robot Controllers in Reinforcement Learning. *IEEE Access*, 2021.
23. Xie J., Zhou R., Liu Y., Luo J., Xie S., Peng Y., Pu H. Reinforcement-Learning-Based Asynchronous Formation Control Scheme for Multiple Unmanned Surface Vehicles. *Applied Sciences 11.2*, 2021. P. 546.
24. Zhou Y., Lu F., Pu G., Ma X., Sun R., Chen Hsi-Yuan, Li X., Wu D. Adaptive leader-follower formation control and obstacle avoidance via deep reinforcement learning. *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE*, 2019.
25. Md Suruz Miah, Amr Elhussein, Keshtkar F., Abouheaf M. Model-free reinforcement learning approach for leader-follower formation using nonholonomic mobile robots. *The Thirty-Third International Flairs Conference*, 2020.
26. Deka A., Luo W., Li H., Lewis M., Sycara K. Hiding Leader's Identity in Leader-Follower Navigation through Multi-Agent Reinforcement Learning. *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE*, 2021.
27. Brockman G., Cheung V., Pettersson L., Schneider J., Schulman J., Tang J., Zaremba W. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.
28. Michel O. Cyberbotics ltd. webots™: professional mobile robot simulation. *International Journal of Advanced Robotic Systems,* 2004, vol. 1, is. 1, pp. 5.
29. Coumans E., Bai Y. *Pybullet, a python module for physics simulation for games, robotics and machine learning,* 2016.
30. Ferigo D., Traversaro S., Metta G., Pucci D. Gym-ignition: Reproducible robotic simulations for reinforcement learning. *2020 IEEE/SICE International Symposium on System Integration (SII), IEEE,* 2020, pp. 885–890.
31. Nathan K., Howard A. Design and use paradigms for gazebo, an open-source multi-robot simulator. *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS),* 2004. (IEEE Cat. No. 04CH37566)
32. Zamora I., Lopez N.G., Vilches V.M., Cordero A.H. Extending the openai gym for robotics: a toolkit for reinforcement learning using ros and gazebo. *arXiv preprint arXiv:1608.05742*, 2016.
33. *S*chulman J., Wolski F., Dhariwal P., Radford A., Klimov O. Proximal policy optimization algorithms // arXiv preprint arXiv:1707.06347v2, 2017.
34. Sutton R.S., Mc Allester D., Singh S., Mansour Y. Policy gradient methods for reinforcement learning with function approximation // Advances in neural information processing systems, 1999, vol. 12.
35. Schulman J., Levine S., Moritz P., Jordan M.I., Abbeel P. Trust region policy optimization. *CoRR, abs/1502.05477*, 2015.
36. Schulman J., Moritz P., Levine S., Jordan M., Abbeel P. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.
37. Liang E., Liaw R., Nishihara R., Moritz P., Fox R., Goldberg K., Stoica I. RLlib: Abstractions for distributed reinforcement learning. In International Conference on Machine Learning, 2018, pp. 3053–3062.
38. A continuous environment for reinforcement learning of the task of the following the leader. Available at: https://github.com/sag111/ContiniousEnvironment_-Follower_Leader (accessed 10.06.2022)