

УДК 004.942

DOI: [10.26102/2310-6018/2026.61.10.019](https://doi.org/10.26102/2310-6018/2026.61.10.019)

Программная платформа имитационного моделирования для обучения и оценки нейросетевых политик автономной навигации наземного беспилотного транспортного средства

Н.М. Горовенко✉, К.В. Богданов

Сибирский федеральный университет, Красноярск, Российская Федерация

Резюме. Актуальность исследования обусловлена необходимостью воспроизводимой разработки и проверки алгоритмов автономной навигации наземных беспилотных транспортных средств до переноса решений на физические устройства. В статье представлена программная платформа имитационного моделирования, объединяющая трехмерную среду, серверный интерфейс управления, веб-интерфейс оператора, средства записи демонстраций, обучения моделей и анализа решений автономного агента. Цель работы состоит в разработке и экспериментальной апробации платформы для исследования нейросетевых политик навигации в лабиринтной среде с частичной наблюдаемостью. Ведущим подходом является имитационное моделирование в трехмерной среде с последующей оценкой алгоритмов обучения по демонстрациям на фиксированных и процедурно генерируемых сценариях. Экспериментальная апробация выполнена на задаче поведения мобильного робототехнического агента, который получает изображение с камеры и данные дальномера. Сравнивались базовая политика поведенческого клонирования и вариант с пространственным контекстом относительно агента. В корректной визуальной среде после повторного обучения базовая политика достигла 10 успешных эпизодов из 30 на сценарии с задержкой управления, а контекстный гибридный вариант – 17 успешных эпизодов из 20 в отдельном базовом демонстрационном сценарии. Дополнительно проведены физические проверки переноса из симуляции на реальное устройство на L- и Z-образных картонных стендах, подтвердившие работоспособность платформенного контура на физическом мобильном роботе и выявившие ограничения исполнительного слоя без точной одометрии. Полученные результаты показывают, что разработанная платформа пригодна не только для демонстрационного запуска моделей, но и для выявления расхождений между метриками на статических данных и поведением агента в замкнутом контуре управления.

Ключевые слова: имитационное моделирование, программная платформа, автономная навигация, наземное беспилотное транспортное средство, обучение по демонстрациям, лабиринтная среда, поведенческое клонирование, визуальный анализ решений.

Для цитирования: Горовенко Н.М., Богданов К.В. Программная платформа имитационного моделирования для обучения и оценки нейросетевых политик автономной навигации наземного беспилотного транспортного средства. *Моделирование, оптимизация и информационные технологии*. 2026;14(10). URL: <https://moitvvt.ru/ru/journal/article?id=2471> DOI: 10.26102/2310-6018/2026.61.10.019

Simulation software platform for training and evaluation of neural-network policies for autonomous navigation of a ground unmanned vehicle

N.M. Gorovenko✉, K.V. Bogdanov

Siberian Federal University, Krasnoyarsk, the Russian Federation

Abstract. The study addresses reproducible development and testing of autonomous navigation algorithms for ground unmanned vehicles before transfer to physical devices. The article presents a

simulation software platform combining a three-dimensional environment, a control server interface, an operator web interface, demonstration recording, model training, and visual analysis of autonomous agent decisions. The purpose of the work is to develop and experimentally validate a platform for studying neural-network navigation policies in a partially observable maze environment. The main approach is simulation modeling in a three-dimensional environment followed by evaluation of learning-from-demonstration algorithms on generated scenarios. The experimental case considers a mobile robotic agent that receives a camera image and range sensor data. A behavior cloning baseline is compared with a variant using agent-centric spatial context. In the correct visual runtime after retraining, the baseline achieved 10 successful episodes out of 30 in the latency scenario, while the contextual hybrid variant achieved 17 successful episodes out of 20 in a basic demonstration scenario. Physical tests of transfer from simulation to a real device were also carried out on L- and Z-shaped cardboard stands: they confirmed the operability of the transfer pipeline on a physical mobile robot and showed that the main reproducibility limitation is the low-level actuation loop without accurate odometry. The results show that the platform is suitable for demonstrating trained models and identifying discrepancies between metrics computed on static data and closed-loop agent behavior.

Keywords: simulation modeling, software platform, autonomous navigation, ground unmanned vehicle, learning from demonstrations, maze environment, behavior cloning, visual analysis of decisions.

For citation: Gorovenko N.M., Bogdanov K.V. Simulation software platform for training and evaluation of neural-network policies for autonomous navigation of a ground unmanned vehicle. *Modeling, Optimization and Information Technology*. 2026;14(10). (In Russ.). URL: <https://moitvvt.ru/ru/journal/article?id=2471> DOI: 10.26102/2310-6018/2026.61.10.019

Введение

Разработка алгоритмов автономной навигации требует среды, в которой можно безопасно повторять сценарии, управлять параметрами эксперимента, записывать данные и сравнивать модели по одинаковым метрикам. При работе только с физическим устройством такие эксперименты становятся дорогими и трудно воспроизводимыми: меняются условия освещения, сцепление, задержки управления, состояние датчиков и начальное положение агента. Поэтому имитационное моделирование остается важным этапом разработки автономных систем и робототехнических приложений [1, 2].

Для задач визуальной навигации особенно важна частичная наблюдаемость. Агент принимает решение по локальным данным: кадру камеры, расстоянию до препятствия, иногда по краткой истории движения. В лабиринтной среде один кадр не всегда однозначно определяет дальнейшее действие. На поведение влияют геометрия коридора, задержка исполнения команды, внешний вид пола и стен, а также то, насколько правильно обучающая выборка отражает будущие сценарии. Поэтому платформа для таких исследований должна позволять не только запускать модель, но и видеть, какой именно вход получает политика и почему она выбирает конкретное действие.

Цель работы – разработать программную платформу имитационного моделирования и экспериментальной оценки нейросетевых политик автономной навигации наземного беспилотного транспортного средства. В рамках цели решались следующие задачи: реализовать связку между трехмерным симулятором и веб-интерфейсом оператора; обеспечить воспроизводимую генерацию лабиринтных трасс; добавить запись демонстраций и обучение политики поведения; реализовать запуск обученной модели в реальном времени; провести экспериментальную апробацию платформы на задаче обучения агента по демонстрациям.

Вклад работы состоит не в утверждении о решении всех вариантов лабиринтной навигации, а в построении воспроизводимого экспериментального контура. Такой контур позволяет проверять гипотезы о качестве моделей, обнаруживать ошибки

визуального домена, сравнивать offline- и online-метрики и фиксировать ограничения алгоритмов. В качестве методической основы использованы глубокие нейросетевые политики [3], behavior cloning [4, 5] и расширения imitation learning [6].

Научно-техническая новизна работы состоит в следующем: 1) предложен воспроизводимый контур оценки навигационных политик, объединяющий генерацию лабиринтных сценариев, запись демонстраций, обучение, online-запуск и визуальный аудит решений в одном цикле; 2) реализована проверка согласованности визуального домена между обучением и исполнением модели; 3) показано расхождение между offline-точностью behavior cloning и online-результатом в симуляторе; 4) *наблюдались разнонаправленные изменения метрик при добавлении ego-центрического пространственного контекста, требующие дальнейшей сценарно-зависимой проверки*¹.

Обзор близких работ. Существующие подходы к разработке автономных агентов можно условно разделить на три группы. Первая группа связана с обучением с подкреплением [1, 2], глубокими нейросетевыми политиками [3] и стандартизированной постановкой задачи взаимодействия агента со средой. Эти работы задают общий аппарат обучения и оценки, но сами по себе не решают задачу согласования данных демонстраций, визуального входа модели и runtime-проверки в конкретной инженерной среде.

Вторая группа относится к обучению по демонстрациям и behavior cloning [4, 5], а также к расширениям imitation learning [6]. Такие методы позволяют строить политику по действиям оператора, однако качество offline-классификации действий не гарантирует успешное поведение при замкнутом управлении агентом, поскольку ошибки политики изменяют последующее распределение состояний.

Третья группа связана с переносом из симуляции и проблемой visual/domain gap [7, 8]. Для таких систем критично фиксировать не только архитектуру модели, но и условия формирования изображения, сенсорные признаки, задержки управления и воспроизводимость сценариев. Современные обзоры sim-to-real переноса подчеркивают, что при переходе к физическому роботу необходимо отдельно учитывать reality gap, параметры симулятора и неопределенность исполнительного слоя [9, 10]. К близким по назначению средам относятся робототехнические и транспортные симуляторы, такие как Gazebo [11], AirSim [12] и CARLA [13]. Они предоставляют богатые возможности моделирования, но в прикладной исследовательской задаче часто требуется отдельный контур, который одновременно связывает запись демонстраций, проверку входа модели, обучение и online-оценку на воспроизводимых учебных сценариях.

В отличие от перечисленных направлений по отдельности, данная работа рассматривает платформу как единый экспериментальный контур: один и тот же стенд используется для генерации маршрутов, записи демонстраций, обучения, online-оценки и визуального анализа входа модели. Это позволяет выявлять не только успешность политики, но и причины расхождения между offline-метриками и поведением агента в симуляторе.

Материалы и методы

Архитектура программной платформы. Платформа включает пять связанных частей: Unity-runtime, серверный интерфейс, веб-интерфейс оператора, модуль обучения и модуль оценки. Unity-runtime отвечает за трехмерную сцену, физику движения, камеру агента, дальномер и телеметрию. Серверный интерфейс обеспечивает обмен командами, получение кадров, управление сценариями и запуск автопилота. Веб-интерфейс

¹ При подготовке выделенных курсивом фрагментов текста использовался генеративный инструмент OpenAI ChatGPT. Модель GPT-5. URL: <https://chatgpt.com/>

используется для подключения к runtime, ручного управления, выбора модели, генерации maze-сценариев и просмотра входа модели. Модуль обучения формирует датасет из демонстраций и обучает политику behavior cloning. Модуль оценки запускает серии эпизодов и сохраняет метрики, checkpoint-артефакты и видео. Выбор Unity-runtime связан с использованием Unity как платформы для построения сред интеллектуальных агентов [8]. Алгоритмическая часть стенда опирается на распространенные реализации методов обучения с подкреплением [14]. Интерфейс взаимодействия со средой согласован с типовой практикой Gymnasium-совместимых окружений [15]. Архитектура программной платформы представлена на Рисунке 1.

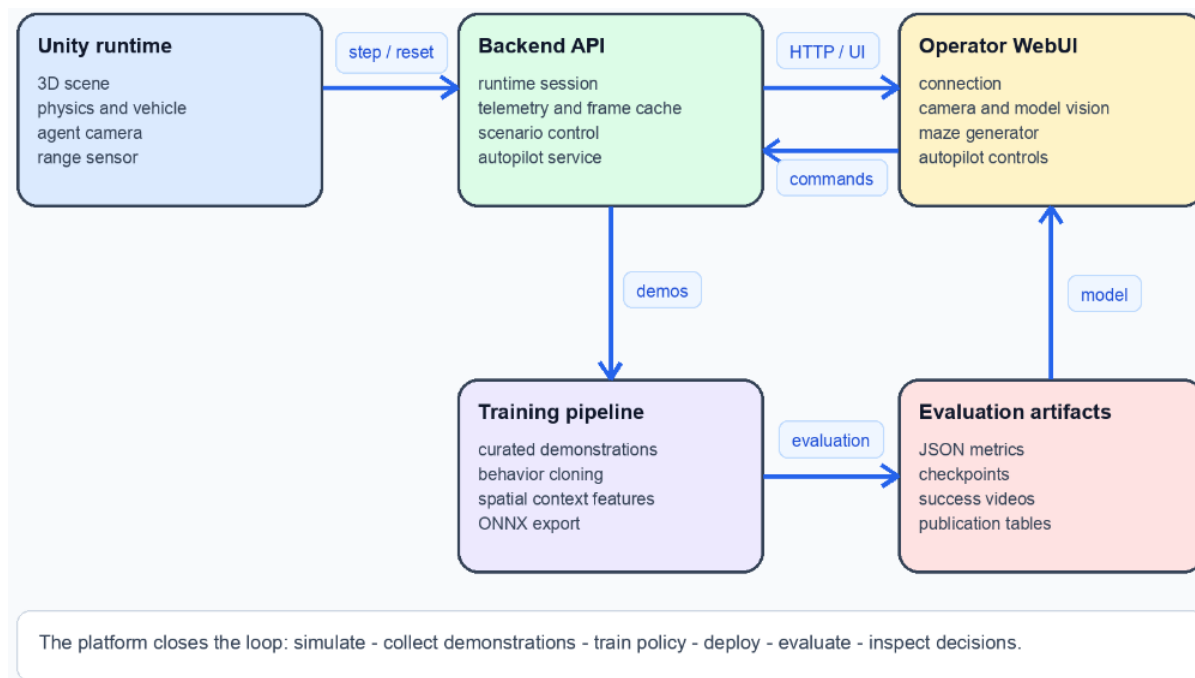


Рисунок 1 – Архитектура программной платформы
Figure 1 – Software platform architecture

Ключевой принцип архитектуры – единый воспроизводимый цикл: симуляция, сбор демонстраций, обучение, развертывание модели, оценка и визуальный анализ решений. Это снижает риск ситуации, когда модель обучается и оценивается в разных визуальных или программных условиях.

Состав основных компонентов платформы приведен в Таблице 1. Такое разделение важно для практического использования стенда: исследователь может менять сценарий или модель, не переписывая физику симулятора, а изменения в WebUI не требуют переобучения политики.

Таблица 1 – Компоненты программной платформы и их назначение
Table 1 – Components of the software platform and their purpose

Компонент	Назначение	Экспериментальный результат
Unity-runtime	Трехмерная сцена, физика, камера агента, дальномер и телеметрия	обеспечивает единые условия запуска эпизодов
HTTP/API слой	Обмен командами, кадрами, состоянием агента и параметрами сценария	позволяет автоматизировать обучение и оценку

Таблица 1 (продолжение)
Table 1 (continued)

WebUI оператора	Подключение к runtime, ручное управление, выбор модели и просмотр телеметрии	снижает стоимость отладки и демонстрации
Maze generator	Создание процедурной трассы по seed и геометрическим параметрам	делает сценарии повторяемыми и расширяемыми
Training pipeline	Подготовка демонстраций, обучение ВС-моделей, экспорт артефактов	связывает пользовательские проезды с политикой управления
Evaluation pipeline	Серийные запуски, JSON-метрики, видео, decision overlay	фиксирует проверяемые свидетельства поведения модели

Эксплуатационный сценарий эксперимента. Работа с платформой строится как повторяемый сценарий исследования. Сначала оператор запускает Unity-runtime и подключает WebUI к выбранному endpoint. После подключения интерфейс получает сведения о доступном транспортном средстве, активном треке, управляющем агенте, выбранной камере и состоянии сенсоров. Затем исследователь может выбрать готовый сценарий или сгенерировать новую maze-трассу по параметрам. Важным свойством является наличие seed: один и тот же набор параметров позволяет восстановить одинаковую геометрию трассы и повторить эксперимент после изменения модели.

На следующем этапе выполняется ручной или автоматизированный проезд. При ручном режиме оператор управляет агентом с клавиатуры или через встроенный control overlay, а система записывает кадры, телеметрию, команды и состояние маршрута. Такой подход позволяет формировать демонстрации без отдельного инструмента разметки. При автоматизированном режиме WebUI связывает выбранную модель с агентом, периодически получает входной кадр, выполняет inference и отправляет дискретную команду в runtime. При этом оператор может переключить основную камеру на вид сверху и одновременно наблюдать вход модели от ego-камеры.

После серии проездов данные передаются в training pipeline. В нем из записанных демонстраций формируются обучающие примеры: изображение камеры, дальность до препятствия, команда оператора, прогресс по маршруту и дополнительные контекстные признаки. Для контекстного варианта строится ego-центрическая осцирапсу тар, которая обновляется по оценке положения агента и дальномеру. Такой формат не требует глобальной карты всей сцены, но дает модели локальную память о ближайшем коридоре.

Финальный этап – серийная оценка. Для каждого варианта политики задается число эпизодов, сценарий, latency-режим и версия runtime. После запуска формируются JSON-отчеты, видеоролики, checkpoint-артефакты и краткие сводки. Такой набор данных удобен для научной работы: численные метрики показывают агрегированный результат, а видео и decision overlay позволяют вручную проверить типичные успешные и неуспешные эпизоды.

Веб-интерфейс и визуальный анализ решения. Для оператора реализован веб-интерфейс, в котором можно выбрать runtime, сцену, агента, камеру, модель и режим автопилота. Отдельная часть интерфейса показывает визуальный вход модели: кадр 84×84, который фактически передается политике, карту локальной памяти, восьмилучевой контекст расстояний и вероятности дискретных действий. Это принципиально отличается от обычного просмотра камеры, потому что оператор может одновременно наблюдать сцену сверху и видеть агентную камеру, используемую моделью. Пример WebUI после подключения к Unity-runtime показан на Рисунке 2. Оператор видит трассу сверху, а ниже отображаются вход модели от ego-агента, карта памяти, контекстные лучи и выбранное действие.

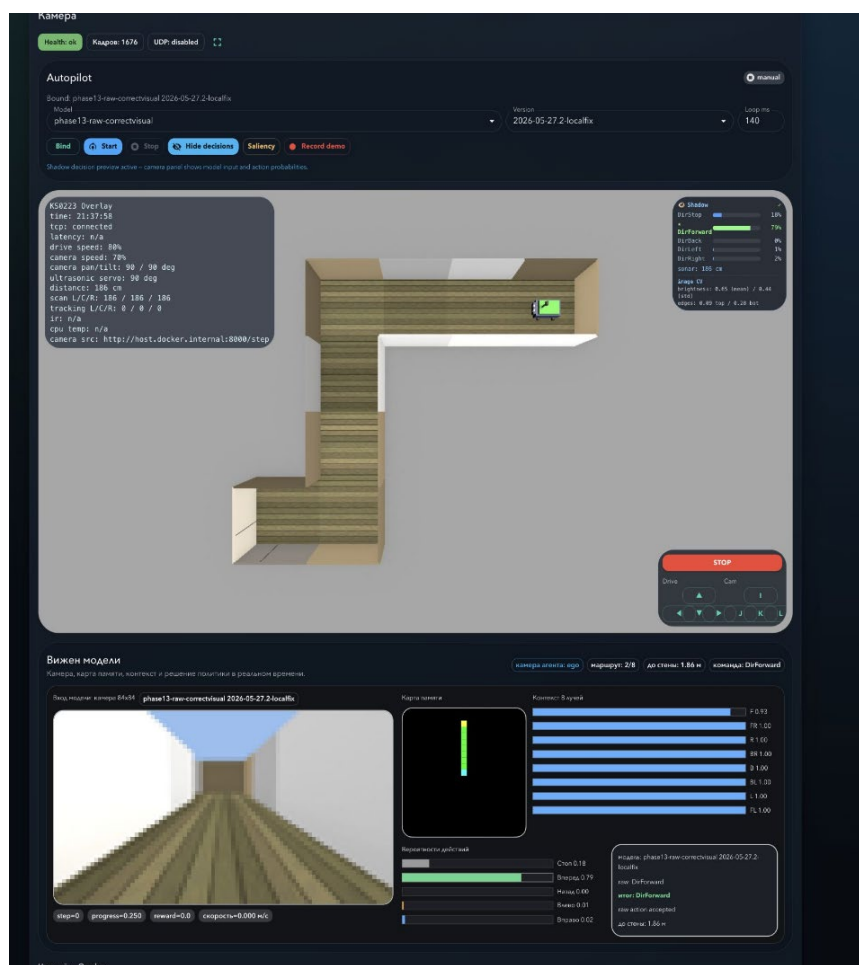


Рисунок 2 – WebUI после подключения к Unity-runtime
Figure 2 – WebUI after connection to Unity-runtime

Такая визуализация оказалась полезной не только для демонстрации, но и для отладки. В ходе работы были выявлены случаи, когда камера оператора и камера модели не совпадали, а также ситуации, когда модель формально выдавала команду движения, но контур безопасности или физический интерфейс не приводил к ожидаемому перемещению.

Для исследовательской работы это имеет отдельное значение. Если оценивать автономный агент только по итоговому флагу успеха, трудно понять причину неудачи: недостаток обучающих данных, неверный кадр камеры, срабатывание защиты, ошибочная карта памяти или плохое обобщение модели. Визуальный слой позволяет сопоставить кадр, телеметрию, вероятности действий и фактическое движение агента в одном интерфейсе. Поэтому WebUI рассматривается не как вспомогательная панель управления, а как часть экспериментального метода.

Лабиринтная среда и данные демонстраций. Экспериментальная среда представляет собой лабиринтную трассу для мобильного агента. Сценарии могут быть фиксированными или процедурно генерируемыми по параметрам: seed, длина маршрута, число левых и правых поворотов, ширина коридора и высота стен. Это позволяет повторять один и тот же эксперимент и расширять набор проверочных условий. При проектировании среды учитывалась проблема расхождения между симуляцией и реальным визуальным доменом, для которой в литературе применяются методы рандомизации домена [7].

Агент управляется пятью дискретными командами: DirStop, DirForward, DirBack, DirLeft, DirRight. Сенсорный вход базовой модели включает изображение с камеры и фронтальное расстояние до препятствия. Для контекстного варианта дополнительно используется его-центрическая карта занятости и вектор расстояний по восьми направлениям.

В качестве обучающих данных использованы десять curated demonstration trajectories, записанных в лабиринтной среде. После подготовки было получено 1378 обучающих примеров. Демонстрации включают разные геометрии маршрутов: L-образные повороты, U-образные участки, S-образную трассу, Z-образную трассу, ступенчатые и длинные извилистые маршруты.

В процессе подготовки данных был исправлен дефект восстановления occupancy map: терминальная клетка стены могла не попадать в канал препятствий при совпадении с предыдущей клеткой луча. После исправления были регенерированы occupancy-массивы для демонстраций. Этот пример важен методически: платформа позволила обнаружить ошибку не в модели, а в данных контекстного представления.

Обучаемые варианты и протокол оценки. В экспериментальной апробации сравнивались три варианта:

1. Raw BC («базовая BC», phase13-raw-retrained) – базовая модель behavior cloning по изображению камеры и дальномеру;
2. контекстная BC-модель с дополнительной occupancy map и distances_8 (Context; «BC + контекст»; phase13-context-retrained-no-rule);
3. контекстная BC-модель с правилом постобработки (Context + rule; «BC + контекст + правило»; phase13-context-retrained-phase8-rule).

Последний вариант является гибридным контроллером, а не полностью нейросетевой политикой. Поэтому его результаты интерпретируются как проверка связки «модель + правило», а не как чистое улучшение нейронной архитектуры.

Основными метриками сравнения были successRate, avgProgress и причины завершения эпизодов: successRate показывает долю успешных завершений, avgProgress отражает средний прогресс вдоль маршрута, а причины завершения позволяют отличить достижение цели от остановки, застревания или тайм-аута. Дополнительно в JSON-отчетах фиксировался avgReward, но в статье он не выносится в основные таблицы, поскольку не дает отдельного вывода сверх доли успехов, прогресса и причин завершения. Главная серия экспериментов выполнена в correct-visual runtime после повторного обучения моделей на тех же демонстрациях.

Единицей статистического наблюдения считался один независимый эпизод запуска политики в заданном сценарии. Для каждого эпизода фиксировались: факт достижения цели, финальный прогресс по маршруту, причина завершения, версия модели, сценарий, режим задержки управления и версия runtime. Для successRate дополнительно указывались абсолютные значения k/N , где k – число успешных эпизодов, N – общее число запусков. Доверительные интервалы для доли успешных эпизодов рассчитывались как 95 % Wilson confidence interval для биномиальной доли. Из-за малых размеров серий интервалы использовались не для доказательства статистического превосходства, а для оценки неопределенности результата. Сравнения имеют описательный характер; статистическая значимость различий между политиками не проверялась. Параметры экспериментального протокола приведены в Таблице 2.

Для воспроизводимости каждый запуск связывался с конкретным сценарием, числом эпизодов, версией модели и режимом runtime. Итоговые данные сохранялись в виде JSON-отчетов, checkpoint-файлов, журналов и видеозаписей с decision overlay. Это позволяет позже проверить не только итоговую строку таблицы, но и отдельные эпизоды: где агент успешно прошел маршрут, где остановился, какой вход получила

модель и какое действие было выбрано. Такой протокол особенно важен для малых экспериментальных серий, где одно среднее значение может скрывать качественно разные типы поведения.

Таблица 2 – Параметры экспериментального протокола

Table 2 – Experimental protocol parameters

Параметр	Значение
Runtime	correct-visual Unity-runtime, локальный HTTP/API endpoint
Обучающие данные	10 curated demonstration trajectories, 1378 обучающих примеров
Вход базовой модели	RGB-кадр 84x84 и фронтальная дальность до препятствия
Вход контекстной модели	RGB-кадр 84x84, дальность, эго-центрическая occupancy map 21x21x3, вектор distances 8
Архитектура	NatureCNN image branch, MLP policy head; для контекстного варианта добавлен compact CNN над occupancy map
Обучение	behavior cloning, 35 эпох, batch size 64, learning rate 3e-4, seed 42
Балансировка классов	для контекстного варианта использован class-balanced sampler
Оценка	deterministic inference, short-latency1: 30 эпизодов с задержкой 1 шаг, seeds 3000-3029; demo01: 20 эпизодов без задержки, seeds 3000-3019
Лимит эпизода	600 шагов; завершения фиксировались как goal, timeout, stuck или stop

Отдельно контролировалось совпадение визуального домена. Первые запуски показали, что модель, обученная и оцененная в разных вариантах визуального окружения, может давать недостоверный вывод о качестве алгоритма. Поэтому финальная серия Phase13 была выполнена после повторного обучения и проверки в одном correct-visual runtime. Этот шаг уменьшает риск интерпретировать техническое расхождение окружений как свойство обучаемой модели.

Артефакты эксперимента хранятся отдельно: исходные демонстрации, восстановленные контекстные признаки, обученные checkpoint-файлы, ONNX-представления, результаты оценки и видеозаписи. Такое разделение упрощает аудит. Если итоговая метрика меняется после исправления ошибки, можно определить, изменилась ли обучающая выборка, модель, runtime или только postprocess-логика. В ходе работы это оказалось практически важным: исправление occupancy map изменило распределение стен в демонстрациях, а исправление источника model-frame устранило ситуацию, когда политика могла получать не тот кадр, который видел оператор.

Таким образом, протокол оценки ориентирован не только на получение максимального successRate, но и на объяснимость результата. Для исследовательской платформы такой протокол важен, поскольку результат должен быть воспроизводимым, проверяемым и пригодным для последующего сравнения.

Результаты

Функционально платформа обеспечивает полный цикл эксперимента: подключение к Unity-runtime, выбор maze-сценария, генерацию трассы по seed, ручное управление, запись данных, обучение модели, экспорт в ONNX, запуск автопилота в WebUI и сохранение артефактов оценки. В результате исследователь получает не только итоговую метрику, но и контекст: видео, вход модели, карту памяти, вероятности действий и JSON-отчеты по каждому запуску. Результаты online-оценки Phase13 приведены в Таблице 3.

Таблица 3 – Результаты online-оценки Phase13 в correct-visual runtime
Table 3 – Phase13 online evaluation results in the correct-visual runtime

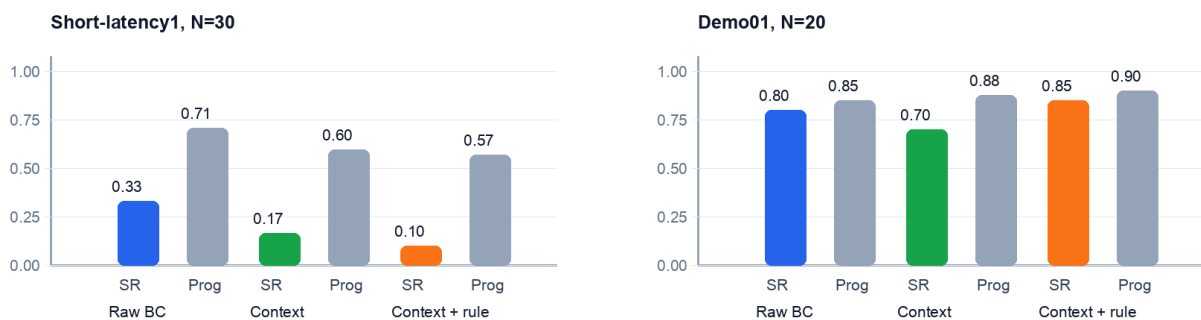
Вариант	Сценарий	k/N	Success rate (95% CI)	Avg. prog.	Интерпретация
Raw BC	short-latency1	10/30	0,3333 (0,192–0,512)	0,7101	наибольшая доля успехов в данной серии с задержкой
Context	short-latency1	5/30	0,1667 (0,073–0,336)	0,5980	ниже Raw BC в данной серии
Context + rule	short-latency1	3/30	0,1000 (0,035–0,256)	0,5695	ниже Raw BC в данной серии
Raw BC	demo01	16/20	0,8000 (0,584–0,919)	0,8526	высокий baseline в данной серии
Context	demo01	14/20	0,7000 (0,481–0,855)	0,8781	выше progress, ниже доля успехов
Context + rule	demo01	17/20	0,8500 (0,640–0,948)	0,9010	наибольшие SR и progress в данной серии

Причины завершения эпизодов приведены в Таблице 4.

Таблица 4 – Причины завершения эпизодов Phase13
Table 4 – Episode termination reasons in Phase13

Вариант	Сценарий	Goal reached	Stalled	Timeout
Raw BC	short-latency1	10	19	1
Context	short-latency1	5	24	1
Context + rule	short-latency1	3	26	1
Raw BC	demo01	16	3	1
Context	demo01	14	6	0
Context + rule	demo01	17	2	1

Сравнение моделей по successRate и avgProgress показано на Рисунке 3.



Interpretation: raw behavior cloning is the strongest correct-visual baseline on the latency scenario; context with rule improves demo01, but does not provide a universal gain.

Рисунок 3 – Сравнение моделей по successRate и avgProgress
Figure 3 – Comparison of models by successRate and avgProgress

На сценарии `short-latency1` наибольшее значение метрики в данной экспериментальной серии показала базовая модель `behavior cloning`: `successRate` = 0,3333, `avgProgress` = 0,7101. Контекстная модель без правила и контекстная модель с `postprocess` показали меньшие значения обеих метрик в данной серии. Этот результат относится к данной серии с задержкой управления.

На сценарии `demo01` картина отличается. Базовая модель достигла `successRate` = 0,8000, а гибридный контекстный вариант – 0,8500; средний прогресс составил соответственно 0,8526 и 0,9010. Наблюдаемое различие в серии `demo01` не следует переносить на произвольные маршруты без расширенной проверки.

Анализ причин завершения показывает, что основным типом отказа остается застревание агента, а не исчерпание лимита времени. На `short-latency1` число застреваний возрастает от 19 у `Raw BC` до 24 у `Context` и 26 у `Context + rule`, что согласуется со снижением `successRate`. На `demo01` контекстный вариант с правилом имеет минимальное число застреваний в серии, однако разница с `Raw BC` соответствует одному эпизоду и требует проверки на большем наборе маршрутов.

Offline-аудит также показал важное расхождение. Ассигасу контекстной модели на демонстрациях составила 0,9434, тогда как у базовой модели – 0,8716. Однако более высокая offline ассигасу не привела к лучшему online-результату на `short-latency1`. Это указывает на необходимость проверки моделей непосредственно в симуляторе, а не только на статическом датасете.

Физический sim-to-real кейс на работе KS0223. В качестве физического `sim-to-real` кейса рассмотрены автономные запуски моделей, указанных в Таблице 5, на работе KS0223 в картонных коридорах. Основной L-стенд имел ширину коридора 0,60 м, первый сегмент 1,10 м, второй сегмент 0,90 м и высоту стен 0,25 м. Дополнительно проверялся Z-образный коридор по схеме: прямой участок, поворот направо, прямой участок, поворот налево, прямой участок. Эти кейсы используются для проверки того, что разработанный платформенный контур способен работать не только в симуляции, но и с реальным устройством, где появляются задержки связи, шум изображения, люфт конструкции, разряд аккумулятора и нестабильность исполнения моторных команд.

Переносу модели на физический стенд предшествовали обучение и проверка в симуляторе, включая прохождение L-образного коридора. К физическим испытаниям переходили после получения удовлетворительных результатов симуляционной проверки. Модель переносилась на робота через единый программный контур; в пределах каждой физической серии модель и настройки управления не менялись. Успехом считалось самостоятельное достижение конечной зоны без касаний стен и вмешательства оператора. Критический сбой означал прерывание заезда из-за контакта со стеной, чрезмерного поворота или нарушения исполнения команд; после касания стены оператор вручную возвращал робота на старт.

Параметры движения в симуляторе настраивались по замерам реального робота на полном заряде: длительность команды сопоставлялась с углом поворота и пройденным расстоянием. Перед заездами проверялись показания и отклик ультразвукового дальномера; визуальный вход в обоих режимах приводился к RGB 84×84. Для `rev29` использовалась доменная рандомизация материалов, текстур пола и освещения. Зачетные заезды старались выполнять на полном заряде, поскольку разряд аккумулятора изменял фактическое движение.

В демонстрационном запуске `rev29` робот самостоятельно прошел прямой участок; после касания стены оператор вручную вернул его в исходное положение. Этот запуск не считается успешным автономным прохождением маршрута. Заезд зафиксирован двумя артефактами: внешним видео длительностью 14,53 с (406×720, 436 кадров) и бортовой камерой длительностью 6,24 с (320×240). В этом контуре `backend-`

autopilot работал с периодом около 140 мс, использовал изображение с бортовой камеры и ультразвуковой дальномер, а safety-стек включал E-stop по сонару, deadman timeout и плавное ограничение тяги.

Для rev29 на симуляционном L-коридоре той же геометрии при расширенной доменной рандомизации получено 15 успешных эпизодов из 20 (75 %). Эти результаты не заменяют оценку моделей Phase14: сопоставимые численные результаты Phase14 на симуляционных L- и Z-коридорах в статье не приводятся.

В новой серии физических запусков проверялось поведение с памятью посещенных участков и поиском прохода вместо безусловной остановки у препятствия. На L-стенде выполнено 20 запусков, из них 8 завершились самостоятельным доездом до финальной зоны, а 12 были сорваны столкновением, чрезмерным поворотом или нестабильностью исполнительного контура. На Z-стенде выполнено 20 запусков, из них 6 завершились самостоятельным доездом до финальной зоны. Наблюдения оператора показывают, что при свободном прямом участке модель выбирала DirForward, а перед препятствием использовала поворот для поиска прохода вместо безусловного DirStop; отдельный счетчик таких локальных ситуаций в этом физическом протоколе не велся, поэтому они не объединяются со статистикой успешности маршрута. По наблюдениям оператора, отказы сопровождалась проблемами исполнительного контура: отсутствием точной одометрии, слишком большим углом поворота, касанием стен, пропусками команд и состоянием аккумулятора.

Таблица 5 – Физический sim-to-real кейс на роботе KS0223

Table 5 – Physical sim-to-real case on the KS0223 robot

Модель	Серия	Объем проверки	Наблюдаемое поведение	Итог
rev29	L-коридор 0,60 м	1 запуск + 56 кадров offline-аудита	baseline-контур camera + sonar + WebUI/autopilot работоспособен	F/L/R/S = 9/47/0/0; Mean P(F) 0,271
phase14	L-коридор	20 запусков	8/20 доездов до финальной зоны	12/20 отказов: столкновение, чрезмерный поворот, исполнительный контур
phase14	Z-коридор	20 запусков	6/20 доездов до финальной зоны	наблюдавшиеся ограничения: одометрия, доставка команд, аккумулятор

Физические картонные L- и Z-коридоры показаны на Рисунке 4.

Эти результаты показывают важное свойство платформы: один и тот же контур позволяет не только запустить модель на физическом устройстве, но и выполнить offline-аудит real-camera кадров, сопоставить решение политики с видеоартефактами и проверить, какие части системы остаются работоспособными при переносе из симуляции на KS0223. При этом модельные ревизии и численные показатели физических запусков зафиксированы в Таблице 5, что отделяет этот результат от симуляционных экспериментов с Phase13.

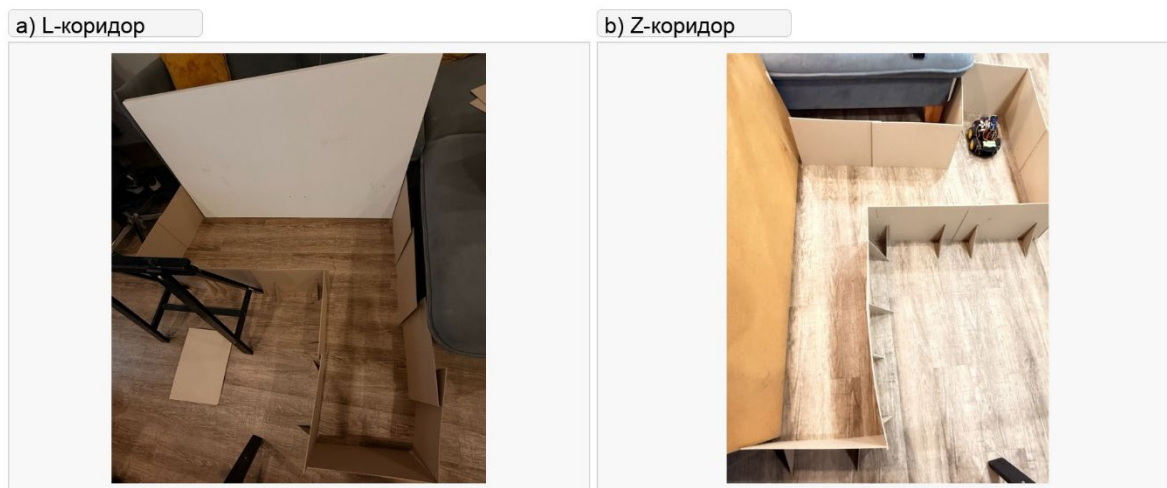


Рисунок 4 – Физические картонные L- и Z-коридоры для sim-to-real проверки KS0223
Figure 4 – Physical cardboard L- and Z-corridors for sim-to-real testing of KS0223

Результаты показывают, что созданный стенд пригоден для проверки и отбраковки экспериментальных гипотез. Если бы оценка ограничивалась только обучающей выборкой, контекстная модель выглядела бы предпочтительнее из-за более высокой offline accuracy. Online-оценка выявила другое поведение: *в одном сценарии наблюдаемые метрики контекстного гибрида выше, в другом ниже, чем у базовой политики*¹. Следовательно, ценность платформы состоит не только в демонстрации успешного запуска, но и в получении отрицательных или частично положительных результатов, которые помогают корректировать дальнейшее направление исследования.

Обсуждение

Полученные результаты демонстрируют две стороны разработанной платформы. С одной стороны, она позволяет быстро собрать данные, обучить модель и проверить ее в управляемой среде. С другой стороны, она помогает выявлять ограничения гипотез. Первоначальная интуиция состояла в том, что пространственный контекст должен улучшать качество навигации в лабиринте. Эксперимент показал более сложную картину: *наблюдаемые метрики при добавлении контекста изменялись разнонаправленно в двух сценариях*¹.

Содержательность результата состоит в том, что платформа выявляет не только успешные, но и отрицательные или частично положительные исходы экспериментов. Платформа не просто демонстрирует заранее удачный заезд, а позволяет воспроизводимо сравнивать варианты, видеть вход модели, фиксировать сбои и уточнять научный вывод. В ходе экспериментов были обнаружены и устранены ошибки визуального домена, ошибки восстановления осциллограмм, проблема различия камеры оператора и камеры модели, а также несоответствие между отображаемой командой и фактическим физическим управлением.

Практическая значимость платформы заключается в том, что ее можно использовать как стенд для дальнейших экспериментов: увеличения числа демонстраций, проверки новых алгоритмов, генерации новых маршрутов, анализа влияния задержек и переноса части решений на физическую модель. При этом текущие результаты не следует интерпретировать как доказательство универсального прохождения произвольных лабиринтов. Наиболее корректная формулировка: демонстрации дают baseline behavior cloning, а spatial context является перспективным, но сценарно-зависимым расширением.

Симуляционные эксперименты Phase13 и физические проверки KS0223 образуют два разных уровня проверки. Симуляционные эксперименты Phase13 использовались для сравнения вариантов нейросетевых политик в контролируемом correct-visual runtime. Физический кейс KS0223 использовался как отдельная проверка переносимости платформенного контура и не рассматривается как статистически сопоставимое продолжение Phase13. Он подтверждает, что связка Unity-runtime, WebUI, Model Registry, backend-autopilot и safety-стек может быть доведена до автономных запусков KS0223 в L- и Z-образных коридорах. Новая серия особенно важна тем, что демонстрирует не только движение вперед на свободном участке, но и поиск прохода перед препятствием: поворот, проверку альтернативного направления и отказ от возврата по уже пройденному пути.

Полученный результат также показывает границу между задачей разработки платформы и задачей построения окончательного алгоритма навигации. Для оценки платформенного вклада важно показать не только работоспособность отдельных модулей, но и замкнутый цикл получения проверяемого экспериментального результата. Для алгоритмической работы следующего этапа потребуется расширять набор демонстраций, проверять другие архитектуры памяти и отказываться от жесткой postprocess-логики там, где она ухудшает обобщение¹.

Платформа формирует связку между 3D-моделированием, веб-интерфейсом, машинным обучением и протоколом оценки. Такая связка может быть использована как учебно-исследовательский инструмент: исследователь получает возможность повторить сценарий, изменить seed трассы, увидеть вход модели и сопоставить решение политики с физическим движением агента.

Ограничения

Исследование выполнено преимущественно в симуляционной среде, поэтому перенос на физическое устройство требует отдельной проверки датчиков, динамики движения, задержек и визуальных условий. Сценарии оценки ограничены двумя основными режимами: short-latency1 и demo01. Количество эпизодов достаточно для предварительного сравнения, но для строгих статистических выводов требуется расширение набора запусков и маршрутов; приведенные доверительные интервалы остаются широкими и не доказывают статистическое превосходство одного варианта политики. Кроме того, вариант Context + rule («ВС + контекст + правило») содержит детерминированное правило, поэтому его нельзя считать чистой нейросетевой политикой.

Физический sim-to-real кейс ограничен двумя картонными трассами, ручным протоколом наблюдений и модельными ревизиями, указанными в Таблице 5. Поэтому он показывает работоспособность платформенного цикла на реальном роботе, но не доказывает универсальное прохождение произвольных maze-сценариев и не должен объединяться со статистикой Phase13 без отдельного протокола сопоставления. Основное техническое ограничение физических запусков связано с отсутствием точной одометрии: одинаковые команды поворота и движения могут давать разные фактические углы и расстояния, особенно при изменении заряда аккумулятора и при пропусках команд.

Еще одно ограничение связано с объемом демонстрационных данных. Использованные curated trajectories дают рабочую основу для behavior cloning, но не покрывают все возможные варианты лабиринтной геометрии, начального положения и визуального шума. Поэтому результаты следует рассматривать как экспериментальную апробацию платформы и baseline-подходов, а не как финальное решение задачи

навигации. В дальнейшей работе целесообразно расширить датасет, добавить кросс-проверку на большем наборе seed и отдельно оценить устойчивость к изменению текстур, освещения и параметров движения.

Заключение

Разработана программная платформа имитационного моделирования для воспроизводимой оценки нейросетевых политик автономной навигации наземного беспилотного транспортного средства в лабиринтной среде с частичной наблюдаемостью. Платформа объединяет Unity-runtime, серверный API, веб-интерфейс оператора, генерацию сценариев по seed, запись демонстраций, обучение behavior cloning моделей, online-запуск политики и визуальный анализ входов и решений модели.

Экспериментальная апробация показала, что offline-точность модели на демонстрациях не является достаточным критерием качества замкнутого управления. В серии short-latency1 базовая BC-модель достигла 10 успешных эпизодов из 30 (successRate=0,3333, 95 % CI 0,192–0,512), тогда как контекстные варианты показали меньшую долю успехов. В серии demo01 гибридный контекстный вариант достиг 17 успешных эпизодов из 20 (successRate=0,8500, 95 % CI 0,640–0,948) и наибольшего среднего прогресса. Из-за ограниченного числа сценариев эти результаты следует интерпретировать как экспериментальную апробацию платформы, а не как доказательство статистически значимого превосходства одной политики¹.

Дальнейшая работа будет направлена на расширение набора сценариев, увеличение числа демонстраций, развитие методов памяти без жесткой rule-based postprocess-логики и расширение физических испытаний обученных политик. Физический кейс показывает, что sim-to-real контур платформы может быть доведен до автономных запусков на KS0223, однако для повышения воспроизводимости требуется улучшить исполнительный слой: добавить одометрию или иной контроль фактического перемещения, стабилизировать доставку команд и учитывать состояние аккумулятора.

СПИСОК ИСТОЧНИКОВ / REFERENCES

1. Sutton R.S., Barto A.G. *Reinforcement Learning: An Introduction*. Cambridge: MIT Press; 2018. 552 p.
2. Kober J., Bagnell J.A., Peters J. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*. 2013;32(11):1238–1274. <https://doi.org/10.1177/0278364913495721>
3. Mnih V., Kavukcuoglu K., Silver D., et al. Human-level control through deep reinforcement learning. *Nature*. 2015;518(7540):529–533. <https://doi.org/10.1038/nature14236>
4. Pomerleau D. ALVINN: An Autonomous Land Vehicle In a Neural Network. In: *Advances in Neural Information Processing Systems, Denver, CO, USA*. Morgan Kaufmann; 1989. P. 305–313.
5. Bojarski M., Del Testa D., Dworakowski D., et al. *End to End Learning for Self-Driving Cars*. arXiv. URL: <https://arxiv.org/abs/1604.07316> [Accessed 30th May 2026].
6. Ross S., Gordon G., Bagnell D. A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning. In: *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics, 11–13 April 2011, Fort Lauderdale, USA*. PMLR; 2011. P. 627–635. URL: <https://proceedings.mlr.press/v15/ross11a.html>
7. Tobin J., Fong R., Ray A., Schneider J., Zaremba W., Abbeel P. *Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World*. arXiv. URL: <https://arxiv.org/abs/1703.06907> [Accessed 30th May 2026].

8. Juliani A., Berges V.-P., Teng E., et al. *Unity: A General Platform for Intelligent Agents*. arXiv. URL: <https://arxiv.org/abs/1809.02627> [Accessed 30th May 2026].
9. Zhao W., Queralta J.P., Westerlund T. Sim-to-Real Transfer in Deep Reinforcement Learning for Robotics: A Survey. In: *2020 IEEE Symposium Series on Computational Intelligence, 01–04 December 2020, Canberra, ACT, Australia*. IEEE; 2020. P. 737–744. <https://doi.org/10.1109/SSCI47803.2020.9308468>
10. Muratore F., Ramos F., Turk G., Yu W., Gienger M., Peters J. Robot Learning From Randomized Simulations: A Review. *Frontiers in Robotics and AI*. 2022;9:799893. <https://doi.org/10.3389/frobt.2022.799893>
11. Koenig N., Howard A. Design and use paradigms for Gazebo, an open-source multi-robot simulator. In: *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems, 28 September – 02 October 2004, Sendai, Japan*. IEEE; 2004. P. 2149–2154. <https://doi.org/10.1109/IROS.2004.1389727>
12. Shah Sh., Dey D., Lovett Ch., Kapoor A. AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles. In: *Field and Service Robotics: Results of the 11th International Conference, 12–15 September 2017, Zurich, Switzerland*. Cham: Springer; 2018. P. 621–635. https://doi.org/10.1007/978-3-319-67361-5_40
13. Dosovitskiy A., Ros G., Codevilla F., Lopez A., Koltun V. CARLA: An Open Urban Driving Simulator. In: *Proceedings of the 1st Annual Conference on Robot Learning, 13–15 November 2017, Mountain View, CA, USA*. PMLR; 2017. P. 1–16. URL: <https://proceedings.mlr.press/v78/dosovitskiy17a.html>
14. Raffin A., Hill A., Gleave A., Kanervisto A., Ernestus M., Dormann N. Stable-Baselines3: Reliable Reinforcement Learning Implementations. *Journal of Machine Learning Research*. 2021;22(268):1–8.
15. Towers M., Kwiatkowski A., Terry J., et al. *Gymnasium: A Standard Interface for Reinforcement Learning Environments*. arXiv. URL: <https://arxiv.org/abs/2407.17032> [Accessed 30th May 2026].

ИНФОРМАЦИЯ ОБ АВТОРАХ / INFORMATION ABOUT THE AUTHORS

Горовенко Никита Максимович, Nikita M. Gorovenko, Master's student at the магистрант Института космических и Institute of Space and Information Technologies, информационных технологий, Сибирский Siberian Federal University, Krasnoyarsk, the федеральный университет, Красноярск, Russian Federation. Российская Федерация.
e-mail: nmgorovenko@yandex.ru
ORCID: [0009-0005-9771-3676](https://orcid.org/0009-0005-9771-3676)

Богданов Константин Валерьевич, Konstantin V. Bogdanov, Candidate of кандидат технических наук, доцент Engineering Sciences, Associate Professor at the Института космических и информационных Institute of Space and Information Technologies, технологий, Сибирский федеральный Siberian Federal University, Krasnoyarsk, the университет, Красноярск, Российская Russian Federation. Федерация.
e-mail: kbogdanov@sfu-kras.ru
ORCID: [0000-0003-2648-1079](https://orcid.org/0000-0003-2648-1079)

Статья поступила в редакцию 16.06.2026; одобрена после рецензирования 08.10.2026; принята к публикации 23.10.2026.

The article was submitted 16.06.2026; approved after reviewing 08.10.2026; accepted for publication 23.10.2026.