DOI 10.37882/2223–2966.2022.06–2.12

РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ СБОРА ДАННЫХ С ДАТЧИКОВ ЛОКОМОТИВА В УСЛОВИЯХ НЕСТАБИЛЬНОЙ ПЕРЕДАЧИ ДАННЫХ

DEVELOPMENT OF A STORAGE SYSTEM USING QUICK DATA ACCESS METHODS

R. Dolmatov
S. Molodyakov

Summary. This paper discusses the issue of creating a simulator for unmanned railway transport, comparing a comparative simulator, the proposed method for compiling a simulator that is different from a significant one, considering the issue of unstable data transfer.

Keywords: unmanned vehicles, simulator, unstable data transfer, devops.

Долматов Роман Александрович
Санкт-Петербургский политехнический университет Петра Великого
d_roman.kst@mail.ru
Молодяков Сергей Александрович
Д.т.н., профессор, Санкт-Петербургский политехнический университет Петра Великого
samolodyakov@mail.ru

Аннотация. В данной работе рассматривается вопрос создания симулятора для беспилотного железнодорожного транспорта, сравнение существующих симуляторов, предложен метод написания симулятора, отличный от существующих, рассмотрен вопрос нестабильной передачи данных.

Ключевые слова: беспилотные транспортные средства, симулятор, нестабильная передача данных, devops.

Введение

Автопилотирование, автомашинист, беспилотное управление — это все разная терминология в определении современных систем автоматизированного управления движением поездов, предназначенного для качественного обеспечения перевозок из одного пункта в другой. Большинство компаний сейчас разрабатывают различные системы автопилотирования и беспилотные транспортные средств: поезда, автомобили, корабли, техники для разной промышленности. Накопленный опыт автоматизации базируется на использовании различных компонентов, начиная от архитектуры процессоров и написания программного кода до современных каналов передачи информации и создания надежной аппаратуры. Эти знания и правильный подход помогает решить проблемы интеграции функций управления движением, безопасностью, совместимость искусственного интеллекта и аппаратной части.

Система автопилотирования позволяет транспортному средству безопасно преодолеть различные расстояния без помощи человека. Беспилотный транспорт — транспортное средство, управление которого осуществляется через специальную систему автономного управления. Данный вид транспорта может передвигаться по заранее запрограммированным маршрутам, либо в общем движении с помощью различных датчиков, включая камеры, радары, лидары, с помощью бортового компьютера. Эффективность системы автоматического управления определяется:

- повышением использования пропускной способности и увеличением провозной способности за счет точного выполнения графика движения
- повышением безопасности движения за счет уменьшения вероятности опасного сближения поездов
- уменьшением затрат энергии поезда за счет выбора оптимальных режимов управления поездом и оптимального по критерию минимума энергозатрат распределения времени хода по линии на время хода по перегонам.

Если мы говорим о понятии беспилотного транспорта, то необходимо упомянуть об уровнях автоматизации транспортных средств. Эти уровни взяты и структуры, установленной профессиональной ассоциацией автомобильных инженеров (SAE). Структура описывает шесть уровней автоматизации транспортных средств, начиная от полного отсутствия автоматизации и заканчивая полностью автоматизированной системой, которая может управлять автомобилем не хуже опытного человека в любой ситуации. В нулевом уровне — водитель-человек несет ответственность за 100% того, что SAE называет «динамической задачей вождения», что означает работу по фактическому вождению транспортного средства на постоянной основе. Уровня 1 — систему, которая обеспечивает либо рулевое управление, либо

Таблица 1. Сравнение трех симуляторов.

Параметр сравнения	Симуляторы		
	CARLA	CA	GAzebo
1	2	3	4
Системные требования	Intel i7 gen 9th — 11th / Intel i9 gen 9th — 11th / AMD ryzen 7 / AMD ryzen 9 +16 GB RAM memory NVIDIA RTX 2070 / NVIDIA RTX 2080 / NVIDIA RTX 3070, NVIDIA RTX 3080 Ubuntu 18.04	Четырёхъядерный процессор с тактовой частотой 4ГГц, Nvidia GTX 1080 8GB	Двухъядерный процессор с тактовой частотой 4 ГГц Nvidia GTX 2080
Операционная система для исполнения	Linux, Windows (64-битная ОС)	Linux, Windows 10 (64-битная ОС) (на Windows работает стабильнее, чем на Linux)	Linux, Windows (64-битная)
Графический движок	Unreal Engine	Unity	Unity 3D
Возможность параллелизации тестов	Присутствует	Отсутствует	Отсутствует
Симуляция физических свойств камеры (добавление шума, автоэкспозиция, блики от объекта, глубина резкости)	Присутствует	Отсутствует	Присутствует
Возможность получения низкоуровневых данных о транспортном средстве (давление и угловая скорость в колесах)	Присутствует	Отсутствует	Отсутствует
Возможность интеграции с фреймворками для автономного вождения	Присутствует	Присутствует	Присутствует
API для различных языков программирования	Python API	Python API	Python API
Наличие камер (RGB, семантика, глубина)	Присутствует	Присутствует	Присутствует
Наличие сенсоров IMU, GPS, лидар, радар	Присутствует	Присутствует	Присутствует
Наличие сенсора	Присутствует	Отсутствует	Присутствует
Наличие датчика столкновения	Присутствует	Отсутствует	Присутствует
Наличие датчика нахождения в полосе	Присутствует	Присутствует	Присутствует
Наличие DVS камеры	Присутствует	Отсутствует	Отсутствует
Количество различных видов транспортных средств	14	4	6
Возможность моделирования собственных сценариев	Присутствует	Присутствует частично	Присутствует

управление ускорением и торможением на постоянной основе, но только при ограниченных, конкретных обстоятельствах. Это не считается «автоматизацией» в рамках SAE, потому что динамическая часть задачи вождения не автоматизирована: человек по-прежнему должен быть готов нажать на тормоз (и деактивировать систему), если впереди более медленное движение. Уровень 2 — «частичная автоматизация». Это для систем помощи водителю, которые обеспечивают как рулевое управление, так и управление ускорением и торможением, но опять же, только при ограниченных обстоятельствах. Если водитель-человек должен регулярно вмешиваться, например, когда автомобиль съезжает с шоссе, то это, вероятно, система уровня 2. Важно отметить, что это не «беспилотное вождение», даже если вроде как кажется, что автомобиль ведет себя сам. SAE определяет уровень 3 как «условную автоматизацию». Разница между уровнем 2 и уровнем 3 заключается в степени. На практике это зависит от ответа на вопрос: насколько внимательным должен быть человек, сидящий за рулем

автомобиля? С системой уровня 2 водитель должен быть очень бдительным, готовым немедленно взять на себя задачу вождения, если система обнаружит что-то, с чем она не может справиться. Ожидается, что на уровне 3 система сможет управлять вождением, пока оно находится в пределах своей «области операционного проектирования», а это означает, что роль человека состоит в том, чтобы быть «запасным вариантом». Уровень 4 — это «высокая автоматизация вождения»: система вообще не нуждается в поддержке человека, пока она работает в своей «области эксплуатационного проектирования». Иными словами, система уровня 4 по-прежнему имеет ограничения, но пока она находится в этих пределах, вмешательство человека не требуется — это настоящее автономное вождение. На практике большинство разрабатываемых сегодня систем «самоуправления» зависят от очень подробных трехмерных карт, которые помогают «мозгу» автомобиля точно знать, где он находится, с точностью до нескольких сантиметров (или меньше). Эти системы обычно используют несколько лидаров для моментального «картографирования» окружения автомобиля. Затем лидарные изображения сравниваются с сохраненной 3D-картой.

Уровень 5 — безусловное (то есть безлимитное) автоматизированное вождение без ожидания вмешательства человека-водителя. Иными словами, система уровня 5 должна быть способна проехать везде, куда может доехать опытный водитель-человек, в любых условиях, с которыми может справиться опытный водитель-человек, полностью самостоятельно. Излишне говорить, что сейчас нет доступных систем уровня 5. Несколько автопроизводителей, в том числе Tesla и BMW, заявили, что в течение нескольких лет у них будут системы 5-го уровня, но многие эксперты считают, что для разработки настоящей автономии 5-го уровня потребуются годы, если это вообще произойдет. В связи с быстрым развитием отрасли беспилотных транспортных средств появляется много инновационных задач. Одна из них — тестирование программного обеспечения. При тестировании необходимо проверить, как будут работать алгоритмы, если на пути беспилотного транспортного средства окажется не только статическое препятствие, но и подвижное, а именно: человек или животное. Также необходимо протестировать алгоритмы в ситуациях, опасных для жизни человека, например, при нарушении пешеходом правил дорожного движения. Большинство компаний использует натурные испытания, во многих случаях возникают проблемы, связанные с безопасностью жизни человека. У компаний, использующих компьютерные технологии, решаются проблемы с помощью симуляторов. Симулятор — приложение, которое воспроизводит реальность и различные сценарии движения транспортного средства, позволяет протестировать различные алгоритмы. Большинство симуляторов созданы для автотранспорта, они помогают развитию сферы автопилотирования. Под нестабильной передачей данных подразумевается задержка пакетов — время, которое необходимо данным, чтобы добраться от источника до «получателя». Совокупная задержка складывается из компонентов:

1. задержка сериализации — время, за которое узел разложит пакет в биты и поместит в линк к следующему узлу. Она определяется скоростью интерфейса.
2. задержка передачи сигнала в среде — результат ограничения скорости распространения электромагнитных волн.
3. задержка обработки пакетов — время на принятие решения и доставку пакета от входного интерфейса до выходного

Исходя из вышеперечисленного можно сделать вывод, что задача разработки симулятора является актуальной задачей, решение которой упростит процесс разработки ПО. Чтобы структурировать всю информацию о трех обозреваемых симуляторах, была составлена сравнительная таблица (табл. 1).

Разработанный симулятор

Движок для симулятора был выбран, исходя из начальных условий. Использовался движок Unity 3D, было решено использовать этот же движок. Для написания клиента был выбран язык программирования Python, так как он очень удобен и легок в применении для данной задачи. Для передачи данных от сервера клиенту был выбран протокол передачи данных protobuf. В процессе создания симулятора осталось объединить наработки двух компаний, добавить необходимых актеров, управляющие воздействие, создать клиент-серверную архитектуру для приема данных и управления симуляцией.

Сервер симулирует мир с актерами, в нем также можно менять погодные условия и время суток. Он также симулирует посылку данных от сенсоров: камер, лидаров, радаров и тепловизора. Также реализована передача истинных значений семантической сегментации и карты глубины. Симулятор включает в себя 3D-модели (пешеходы, животные, велосипедисты, деревья и др.) и сценарий поведения этих объектов. Есть возможность моделировать поведение людей и животных и тестировать реагирование алгоритмов на появление препятствий в разных зонах. Клиент же посылает серверу сигналы управляющего воздействия: включение/выключение фар, подача гудка, экстренное торможение. Физика разгона и торможения состава тоже вынесена в клиентскую часть, данные берутся из протоколов испытаний,

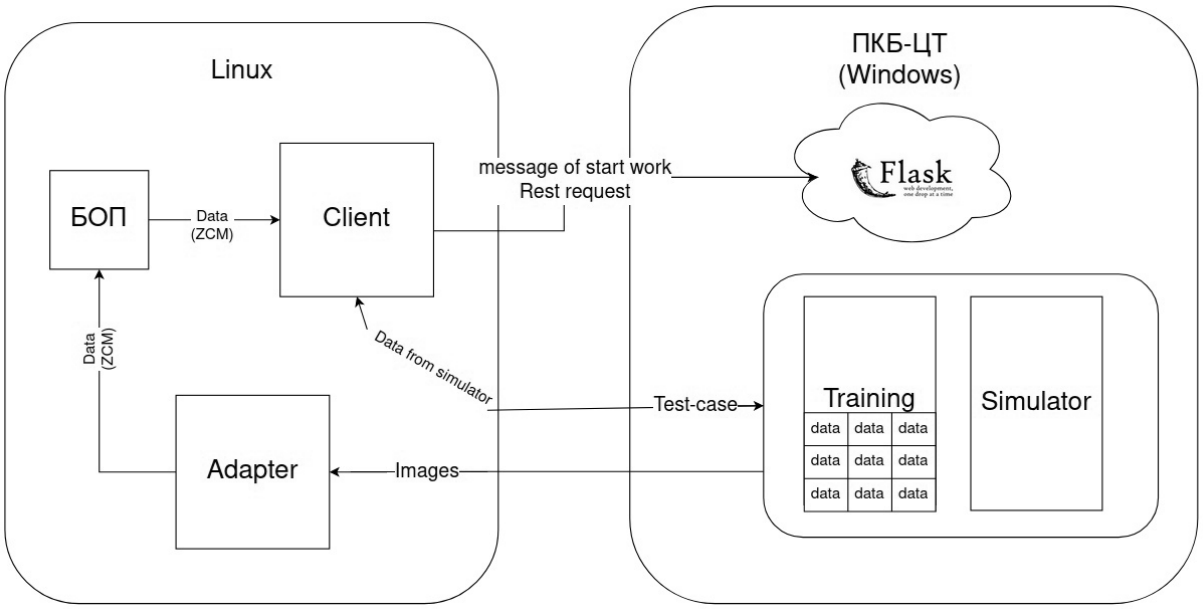

Рис. 1. Схема работы симулятора

а на сервере скорость задается константой. Вся схема представлена на Рис. 1

Основные компоненты симулятора:

- Симулятор, расположенный на операционной системе Windows.
- Программное обеспечение, развернутое на OS Linux, обеспечивающее работу тренажера и подключение к симулятору

На операционной системе Windows запускается сервер Flask, после чего запускается программное обеспечение на Linux. Так как с сервером связан клиентский компонент, то запускается именно он. Клиент отправляет сообщение в сервер о начале работе. Сервер после получения сигнала запускает одновременно симулятор и тренажер. Следующий компонент, с которым связан симулятор — адаптер. Работающий симулятор с включенной камерой отправляет кадры адаптеру, после чего идет преобразование в zcm-формат. Клиент отправляет в симулятор данные из тест-кейсов и принимает данные из симулятора и следит, прошел тест-кейс или нет. Когда все тест-кейсы пройдены, то клиент посылает сообщение на сервер, чтобы тот закончил работу симулятора. Все будет выполняться до тех пор, пока все вариации тест-кейсов не закончатся.

Внедрение симулятора

Итак, этапы внедрения симулятора:

1. Проверка работоспособности симулятора в целом
2. Тестирование без препятствий
3. Тестирование с настоящими препятствиями
4. Тестирование редких сценариев
5. Ввод в эксплуатацию

Каждый этап будет рассмотрен подробнее.

На первом этапе внедрения симулятора происходит тестирование данной технологии. Проверяют отправку и получение данных, управляющие воздействие, работу сенсоров, изменение погодных условий и времени суток, включение прожектора, т.е. тестируют симулятор в целом, отдельно от ПО. На данном этапе внедрения симулятора не заменяется ни одно натурное испытание.

На втором этапе симулятор проходит проверку на ложные срабатывания, а именно: проверяется его работоспособность без препятствий. Данный этап поможет исключить срабатывание алгоритма остановки в ситуациях, когда препятствий нет. На данном этапе мы можем заменить 10% натурных испытаний, ПО уже можно тестировать в различных условиях, но без препятствий.

Далее, идет проверка симулятора на работу с настоящими препятствиями, здесь проверяется отработка разного вида торможений. На данном этапе можно заменить 58% натурных испытаний. И при необходимости с помощью симулятора уже можно собирать различные датасеты.

Последний этап до ввода симулятора в постоянную эксплуатацию — тестирование редких сценариев. Про-

исходит проверка возможности создания необходимых сценариев и их работоспособности. Здесь процент натурных испытаний остается таким же, как на предыдущем этапе.

После проведения всех проверок симулятор вводят в эксплуатацию, на данном этапе можно заменить 70% натурных испытаний тестированием ПО с помощью симулятора.

Понятие нестабильной передачи данных

Для управления информационными пакетами, передаваемыми по сети, разработана технология Quality of Service — это набор технологий, которые запускают высокоприоритетные приложения и трафик при лимитированной пропускной способности. Это означает, что более важный трафик будет обработан быстрее, а задержки по сети будут минимальны. Необходимо разделить трафик с помощью инструментов классификации. Так организации смогут контролировать доступность ресурсов для приоритетных приложений. Потери говорит о том, сколько из отправленных источником пакетов дошло до адресата. Причиной потерь может быть проблема в интерфейсе/кабеле, перегрузка сети, битовые ошибки, блокирующие правила ACL. Что делать в случае потерь решает приложение. Оно может проигнорировать их, как в случае с телефонным разговором, где запоздавший пакет уже не нужен, или перезапросить его отправку — так делает TCP, чтобы гарантировать точную доставку исходных данных.

Задержки — это время, которое необходимо данным, чтобы добраться от источника до получателя. Совокупная задержка складывается из следующих компонентов.

- Задержка сериализации (Serialization Delay) — время, за которое узел разложит пакет в биты и поместит в линк к следующему узлу. Она определяется скоростью интерфейса. Так, например, передача пакета размером 1500 байтов через интерфейс 100Мб/с займёт 0,0001 с, а на 56 кб/с — 0,2 с.
- Задержка передачи сигнала в среде (Propagation Delay) — результат ограничения скорости распространения электромагнитных волн. Физика не позволяет добраться из Нью-Йорка до Томска по поверхности планеты быстрее чем за 30 мс (фактически порядка 70 мс).
- Задержки, вносимые QoS — это томление пакетов в очередях (Queuing Delay) и последствия шейпинга (Shaping Delay).
- Задержка обработки пакетов (Processing Delay) — время на принятие решения, что делать с пакетом: lookup, ACL, NAT, DPI — и доставку его от входного интерфейса до выходного.

Заключение

В рамках данной работы был рассмотрен проект создания симулятора для беспилотного железнодорожного транспорта. В ходе работы были изучены и проанализированы симуляторы для автомобильного беспилотного транспорта, была подробно рассмотрена их архитектура, наполнение и способы применения, понятие нестабильной передачи данных.

ЛИТЕРАТУРА

1. Статья «История беспилотных автомобилей» [Электронный ресурс] URL: https://bespilot.com/info/istoriya;
2. John Rosevear, «Self-Driving Cars: Understanding the 6 Autonomous Levels», 2018
3. J. Banks; J. Carson; B. Nelson; D. Nicol (2001). Discrete-Event System Simulation. Prentice Hall. p. 3. ISBN978–0–13–088702–3;
4. A. Dosovitskiy, G. Ros, F. Codevilla, A. López and V. Koltun, «CARLA: An open urban driving simulator», Conference on Robot Learning (CoRL), 2017
5. A. Dosovitskiy «CARLA Documentation». [Электронный ресурс]: CARLA Simulator. URL: https://carla.readthedocs.io/en/latest/start_introduction/;
6. Rong, Guodong and Shin, Byung Hyun and Tabatabaee, Hadi and Lu, Qiang and Lemke, «LGSVL Simulator: A High Fidelity Simulator for Autonomous Driving», arXiv preprint, 2020;
7. Ermakov N.V., Molodyakov S.A. A caching model for a quick file access system // Journal of Physics: Conference Series. IOP Publishing, 2021. Т. 1864. № . 1.

Журнал «Современная наука: актуальные проблемы теории и практики»