МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Факультет компьютерных технологий и прикладной математики**

**Кафедра вычислительных технологий**

Допустить к защите

Заведующий кафедрой,

д-р техн. наук, проф.,

\_\_\_\_\_\_\_\_\_ А.А. Еремин

(подпись)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2024г.

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

**(БАКАЛАВРСКАЯ РАБОТА)**

**РАЗРАБОТКА СИМУЛЯТОРА АВТОПИЛОТА ТРАМВАЯ**

Работу выполнил \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Д.Е. Гиренко

(подпись)

Направление подготовки 02.03.02 — «Фундаментальная информатика и информационные технологии»

Направленность (профиль) «Вычислительные технологии»

Научный руководитель,

канд. техн. наук, доц.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Т.А. Приходько

(подпись)

Нормоконтролер,

канд. техн. наук, доц.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Е.А. Нигодин

(подпись)

Краснодар

2024

**РЕФЕРАТ**

Выпускная квалификационная работа 53 стр., 9 ч., 58 рис., 16 формул, 12 источников, 4 приложения.

АВТОПИЛОТ, YOLO, НЕЙРОННЫЕ СЕТИ, СИМУЛЯТОРЫ, РАБОТА С ПАМЯТЬЮ ПРОЦЕССА, ТРАМВАЙ

Объектом исследования являются архитектура и особенности разработки автопилота трамвая для симулятора.

Предметом исследования в данной работе являются функциональность автопилота, его автономность и эффективность для симуляции, а также реализация оптимальных методов общения автопилота и симулятора.

Цель работы: разработка программы автопилота для управления трамваем в симуляторе.

Методологическая основа включает в себя анализ литературы и документации, сравнительный анализ различных подходов и практик, математическое и программное моделирование, экспериментальное тестирование различных методов и практик в разработке автопилотов, формулирование выводов.

Научная новизна работы заключается в изучении возможности использования автопилота для управления трамваем в симуляторе, а также выкладывание результатов исследования в общий доступ.

В результате решения поставленных задач была реализована программа автопилота для симулятора управления трамваем TramSim Vienna, использующая YOVLv8 для детекции объекты с видеопотока и получающая значения скорости трамвая напрямую с памяти процесса, которая способна проходить маршрут трамвая без столкновений, производить посадку-высадку пассажиров на каждой остановке и следовать сигналам светофоров.

**СОДЕРЖАНИЕ**

[Введение 5](#_Toc104743664)

[1 Формальная постановка задачи определения профориентации 7](#_Toc104743665)

[2 Теоретические основы Data Mining текстовых данных 8](#_Toc104743666)

[2.1 Понятие Data Mining 8](#_Toc104743667)

[2.2 Методы классификации тональности 8](#_Toc104743668)

[2.2.1 Методы, основанные на правилах и словарях 8](#_Toc104743669)

[2.2.2 Машинное обучение с учителем 9](#_Toc104743670)

[3 Теоретические основы нечеткого анализа 9](#_Toc104743671)

[3.1 Понятие лингвистической переменной и терма 9](#_Toc104743672)

[3.2 Построение функции принадлежности косвенным методом опроса Т. Саати 10](#_Toc104743673)

[3.3 Построение функции принадлежности косвенным методом опроса А.П. Ротштейна 11](#_Toc104743674)

[4 Рассматриваемая классификация профессий 14](#_Toc104743675)

[5 Разработка социальной сети 17](#_Toc104743676)

[5.1 Предпосылки разработки социальной сети 17](#_Toc104743677)

[5.2 Программная реализация серверной части социальной сети 17](#_Toc104743678)

[5.3 Программная реализация клиентской части социальной сети 18](#_Toc104743679)

[6 Разработка системы Data Mining 21](#_Toc104743680)

[6.1 Инструменты анализа данных 21](#_Toc104743681)

[6.2 Набор данных для обучения моделей 21](#_Toc104743682)

[6.2.1 Набор данных для обучения моделей классификации тональности текста 21](#_Toc104743683)

[6.2.2 Набор данных для обучения моделей классификации тем документов 22](#_Toc104743684)

[6.3 Разработка системы классификации тональности текста 23](#_Toc104743685)

[6.4 Разработка системы классификации тем документов 34](#_Toc104743686)

[7 Разработка экспертной системы 38](#_Toc104743687)

[7.1 Построение экспертной системы 38](#_Toc104743688)

[7.2 Сбор оценок экспертов 39](#_Toc104743689)

[7.3 Реализация и тестирование экспертной системы 39](#_Toc104743690)

[8 Схема работы системы прогнозирования 43](#_Toc104743691)

[9 Тестирование системы прогнозирования профориентации 44](#_Toc104743692)

[Заключение 51](#_Toc104743693)

[Список использованных источников 52](#_Toc104743694)

# **ВВЕДЕНИЕ**

В современном мире технологические достижения стремительно преобразуют нашу жизнь. Существенный прогресс в области нейросетей открывает перед нами возможности создания сложных алгоритмов, способных автоматизировать процессы управления транспортными средствами, включая трамваи.

Трамваи играют значительную роль в системе общественного транспорта России, обеспечивая массовый перевоз пассажиров в крупных городах. Их эффективное и безопасное функционирование не только важно для плавного движения городской жизни, но и неразрывно связано с общественным комфортом и экономическим развитием регионов. Вместе с тем, несмотря на значимость трамвайного транспорта, существует потребность в постоянном движении вперёд, освобождая людей от рутинной работы и повышая уровень безопасности и эффективности транспортной системы. В этом контексте разработка автопилота для трамваев выступает как важное направление инноваций, направленных на улучшение жизни городского населения и оптимизацию транспортных процессов.

Целью данной работы является изучение и реализация подходов написания автопилота для симулятора управления трамваем для того, чтобы на практике оценить весь объём предметной области и столкнуться с любыми возможными проблемами разработки, чтобы решить.

Данная работа состоит из 7 глав.

Первая глава содержит теоретические сведения касательно автопилотов и известные на данный момент знания о программном управлении трамваем.

Вторая глава содержит описание выбранного для работы симулятора.

Третья глава содержит анализ и методы решения проблемы подключения потоков входной информации и выходных сигналов управления.

В четвертой главе рассматривается вся работа с нейронной сетью, решающей задачу детекции объектов на изображении.

В пятой главе описывается разработка модуля принятия решения.

В шестой главе описывается итоговая система, объединяющая все реализованные решения.

В седьмой главе проводится тестирование разработанной системы на маршруте в симуляторе.

# **1 Общая теория автопилотов**

## **1.1 Определение и использование автопилота в транспорте**

Автопилот — программно-аппаратный комплекс, управляющий транспортным средством, для следования определённому маршруту или выполнения особых задач. То есть система автопилотирования – это не только программный код, вычисляющий правильную последовательность действий для транспорта, но и ещё весь набор датчиков, позволяющих получать необходимую информацию, и устройства вывода, передающие цифровые сигналы в механические части транспорта. Среди автопилотов наземного транспорта, зачастую, такой комплекс устанавливается в уже готовый транспорт как дополнение в виде камер, спутникового приёмника, лидара, радара, компьютера и проводов, соединяющих всё.

Стоит ещё отметить, что не все транспортные средства способны подключиться к автопилоту, а только те, которые обладают CAN-шиной — часть системы управления автомобилем. Система использует двоичные коды, чтобы обмениваться информацией между различными частями транспортного средства. Весь процесс протекает в реальном времени, без задержек, а также отличается высочайшей надёжностью. В 2018 году использование CAN-шины в разъёме для диагностики стало обязательным условием для автомобилей, которые продаются в США, так что использование этой шины не только в личном транспорте, но и в любой технике уже давно является стандартом.

Большинство людей знают об автопилотах из авиации, где они используются повсеместно. По больше части причина этому то, что в воздушном пространстве самолёту достаточно выполнять вполне простые задачи: лететь по заданному маршруту и держать судно в правильно положении. С этим могут справиться даже обычные программы с математической логикой. Однако, в наземном транспорте всё становится сложнее в виду множества различных объектов, которые могут встретиться на дороге и на которые необходимо правильно реагировать, в том числе и другие участники дорожного движения. Причём, чем меньше масштаб, тем большая точность требуется от управляющего комплекса. Если для большегрузного автомобиля с задачей доставки груза через несколько километров по территории закрытого склада ещё можно воспользоваться только навигацией с GPS сигнала, то для параллельной парковки личного автомобиля компьютеру понадобится точное понимание расстояния до объектов, для чего используют радар или даже лидар.

## **1.2 Определение автопилота по уровням**

Как часто бывает в технике для того, чтобы достичь задуманного решения необходимо наметить путь совершенствования решения и решать задачи последовательно. Аналогично и среди автопилотов выделяют пять уровней:

1. Водительская помощь. На уровне 1 присутствует частичная автоматизация, при которой система может управлять либо ускорением и торможением, либо рулевым управлением, но не обеими функциями одновременно. Примером может служить адаптивный круиз-контроль, который поддерживает заданную скорость и дистанцию до впереди идущего автомобиля, требуя от водителя управления рулем;
2. Частичная автоматизация. Предполагает наличие системы, которая способна одновременно управлять и ускорением/торможением, и рулевым управлением. При этом водитель обязан постоянно контролировать дорогу и быть готовым взять управление в любой момент;
3. Условная автоматизация. Система управления транспортным средством способна самостоятельно выполнять все функции вождения в определенных условиях и средах, таких как шоссе. Водитель может не следить за дорогой, но должен быть готов вмешаться по запросу системы;
4. Высокая автоматизация. Система способна самостоятельно управлять транспортом в большинстве ситуаций без вмешательства водителя. Водитель может полностью отвлечься от управления в течение всего времени движения в пределах определенных зон или условий, например, в городских районах с низкой скоростью или на определенных маршрутах. Если система сталкивается с условиями, которые она не может обработать, она может безопасно остановить транспорт;
5. Полная автоматизация. Транспорт способен выполнять все задачи по управлению транспортным средством в любых условиях и на любых дорогах без какого-либо вмешательства со стороны человека. Водитель становится пассажиром и не требуется вообще для управления автомобилем. На данном уровне могут отсутствовать даже традиционные элементы управления, такие как педали и рулевое колесо. Более наглядный примера можно посмотреть на рисунок 1.



Рисунок 1 – Уровни автономности.

Так что сейчас в автопилотировании стоит вопрос о том, как создать автопилот, соблюдающий все правила дорожного движения и способного разумно реагировать на любые непредвиденные ситуации в пути, подобно человеку. Впрочем, в последние годы отличные результаты показывают автопилот от компании Tesla, использующий только камеры для управления автомобилем, а также автопилотные такси от российской компании Яндекс. Однако их решение нельзя назвать полным автопилотом в виду технических ограничений. Кроме того, ни одна из компаний не стремится к созданию полного автопилота для дорог общего пользования в виду того, что ни одно законодательство мира не спешит определять законы для транспортного средства, которое работает на автопилоте. Для решения юридического вопроса в той же России необходимо будет много времени и сил со стороны компаний-производителей автопилотов. Но помимо этой преграды, автопилотам ещё придётся пройти через недоверие людей, которые не захотят доверять свою жизнь машине.

## **1.3 Особенности разработки автопилота для трамвая**

В некоторых странах, помимо самолётов, ещё и поезда используют автопилот или автоматизированные системы управления, так как задач, которые должен выполнять водитель, например, грузового поезда, намного меньше. Поэтому трамвай — это идеальная «песочница» для полного автопилота автомобиля, так как он совмещает в себе, как и облегчения от вождения поезда, так и должен соблюдать правила дорожного движения и участвовать во взаимодействии с пешеходами и автомобилями.

Трамвайный маршрут всегда известен, транспортное средство не может с него сойти. Кроме того, маршрут, зачастую, регулярный, а значит можно проехать заранее и разметить точки со светофорами и прочим, чтобы системе было легче их распознавать. Трамвай не может перестроиться из полосы в полосу, что является огромной нагрузкой автопилота автомобиля, а здесь же и вовсе отсутствуют повороты. Кроме того, тормоз за счёт железнодорожного полотна работает очень резко и способен остановить трамвай практически мгновенно. А с другой же стороны, для трамвая добавляется задача особого поведения на остановках, чтобы производить посадку и высадку пассажиров.

Благодаря всем особенностям управления, трамвай имеет потенциал стать первым общественным транспортом, который сможет полностью перейти на автоматическое пилотирование или хотя бы заменить водителя на бортинженера, что значительно уменьшит кадровый голод этой профессии. В данный момент реализацией этой идеи занимается дочерняя компания Сбербанка – Cognitive Pilot. У них уже имеется успешный проект по написанию автопилота для сельскохозяйственных комбайнов, знания из которого они и применяют сейчас экспериментальных автопилотов, которые уже тестируются на дорогах Петербурга. За пару лет жизни проекта компания смогла представить статистику, доказывающую эффективность использования автопилота перед ручным управлением. За счёт строго выполнения правил и машинной точности удалось значительно увеличить безопасность поездки для пассажиров и уменьшить износ техники и железнодорожных путей, несмотря на неприязнь к новой технологии водительского состава.

Именно этот проект вдохновил меня на создание подобного автопилота для управления трамваем. В виду того, что ни одно депо не выделило бы мне для написания работы реальный трамвай, было решено использовать симуляцию этого транспортного средства.

# **2 Описание симулятора TramSim**

Задача написания автопилота для реального транспортного средства совсем не тривиальная задач в виду сложности моделирования всех возможных физических процессов и событий в компьютерной среде. В виду ограниченности ресурсов, было решено использовать симулятор трамвая. Он, конечно, создаёт намного меньше реальных проблем, с которыми сталкиваются команды разработки на реальной технике, но, с другой стороны, симулятор и сильно ограничен по своим программным возможностям, например, полным отсутствием GPS, удобной CAN-шины и многим другим. Так что предстояло решать сходные проблемы, но в цифровой среде, с расчётом на то, что созданная мной система может быть использована в будущем на реальном транспорте при минимальных изменениях модулей получения информации и передачи выходного сигнала.

Для создания симулятора работы трамвая в городском окружении было решено использовать компьютерную игру TramSim Vienna [2]. Основным критерием выбора стала разработка на игровом движке Unreal Engine, что обеспечивает высокое качество графики и приближает визуальное восприятие к реальности. Этот выбор позволяет пользователям получить максимально реалистичный опыт управления трамваем, воспроизводящий условия работы настоящего транспортного средства в городской среде.

Одним из важных преимуществ TramSim Vienna является опыт разработчиков в создании симуляторов различных транспортных средств. Их аккуратность и внимание к деталям отражаются в реализации транспортного средства в игре, а также в его отзывчивости к действиям игрока. Разработчики серии игр TramSim обладают значительным опытом, который позволяет им создавать симуляторы, максимально приближенные к реальной работе транспортных средств, что делает выбор TramSim Vienna логичным и обоснованным для целей данного исследования. Пример графики можно увидеть на рисунке 2.



Рисунок 2 – Пример графики из TramSim Vienna.

В данной игре можно полностью проделать основную работу водителя трамвая на определённом маршруте, а именно следовать ему согласно правилам дорожного движения, правильно реагировать на сигналы светофоров и производить посадку и высадку пассажиров на остановках. В симуляторе присутствуют машины, имитирующие реальный трафик, а также пешеходы, которые могут приходить на остановку и перемещаться на трамвае.

# **3 Налаживание потоков ввода и вывода**

## **3.1 Установка камеры и захват изображения**

В контексте реальных систем сбора информации о внешнем мире, обычно используются разнообразные датчики, такие как камеры, радары, лидары, а также информация с датчиков, подключенных к CAN-шине трамвая. Однако, в случае с симулятором TramSim Vienna, ограничения на доступ к исходному коду не позволяют модифицировать транспортное средство или добавить новые сенсоры для сбора данных. Из-за этого, единственный способ получения информации о находящихся вокруг трамвая объектов являются виды: от первого лица, из кабины, свободная камера в системе координат трамвая и полностью свободная камера. Переключение между ними осуществляется при помощи горячих клавиш, причём мгновенно и с сохранением положения, что крайне удобно.

Вид от первого лица водителя не подходит, так как большая часть обзора закрыта приборной панелью и кабиной трамвая. Пример такого кадра можно увидеть на рисунке 3. Чтобы не терять информацию, было решено использовать свободную камеру, которая следует за трамваем, установив её на носу транспорта. Для того, чтобы при каждом запуске камера занимала одно и то же место, был написан специальный скрипт, эмулирующий движения курсором. Остальные же виды было решено использовать только в случае необходимости.



Рисунок 3 – Вид из кабины.

Теперь, когда основная камера находится на своём месте необходимо получить с неё картинку в программу автопилота. Для получения потока видео из игры, можно было воспользоваться специальными программами записи, но для оптимальности было решено воспользоваться готовым API от Windows, которым можно было воспользоваться в одноимённой библиотеке Python-а «windows\_capture». Достаточно было указать имя открывающегося приложения.

## **3.2 Получение скорости и ускорения**

В реальных системах автопилота получение скорости и многих других данных – тривиальная задача. Достаточно подключить компьютер к CAN-шине и начать читать поток. Однако внутри симулятора такой метод решения задачи недоступен.

Первоначально были предприняты попытки с помощью игровой консоли выводить на экран число, а после его считывать вместе со всей картинкой. Однако, все попытки найти подобную команду, даже прямое письмо разработчикам игры с просьбой помочь, не увенчались успехом. Тогда появилась идея о том, чтобы использовать возможность быстрого переключения между камерой из кабины водителя и свободной камеры на носу трамвая для того, чтобы разделить все считываемые кадры игры на кадры с изображением окружающей местности впереди и на кадры с изображением спидометра. Для перевода кадра, содержащего спидометр, в вещественное число скорости трамвая использовалась функция, что из всего кадра вырезает отдельный прямоугольник со спидометром, после чего подсчитывает количество пикселей зелёного (в цвет полоски спидометра) цвета. Однако этот метод имел крайне большую ошибку и не мог использоваться для стабильного использования. Чтобы добиться меньшей ошибки, пришлось бы использовать ещё одну нейросеть, определяющую по картинке спидометра, какую скорость он показывает, или же создать ещё больше эвристик для подсчёта пикселей.

Тогда было решено, по аналогии с CAN-шиной, получить доступ к памяти процесса игры на прямую и оттуда начать получать значение скорости движения трамвая. Для работы с памятью процесса и получения нужного адреса значения было решено воспользоваться Cheat Engine.

Cheat Engine — это программа с открытым исходным кодом, которая позволяет пользователям изменять параметры в компьютерных играх. Она обеспечивает возможность изменения числовых значений. Пользователи могут сканировать память игры, находить адреса и значения, которые могут быть изменены, и создавать чит-таблицы для использования в игре. В основном программу применяют для редактирования числовых значений в видеоиграх для получения каких-либо нечестных бонусов. Аналогом Cheat Engine является и Art Money, однако за счёт долголетия жизни проекта в качестве программы с открытым исходным кодом, функционал Cheat Engine опережает Art Money.

Примером такого расширенного функционала являются инструменты для автоматизации определенных действий в игре через запись и воспроизведение скриптов, а также получение информации о постоянных указателях на те или иные значения в памяти. Именно этими функциями и предстояло воспользоваться, чтобы прямо с процесса работы симулятора получать информацию о скорости.

Операция получения постоянного адреса скорости трамвая с помощью Cheat Engine состояла из нескольких шагов:

1. Из всего множества адресов памяти, которые использует процесс «TramSimVienna-Win64-Shipping.exe» выделить тот, который хранит в себе значение скорости трамвая. Адрес находился достаточно тривиально – после ускорения трамвая в игре адреса отсеивались по признаку увеличения значения, хранимого по адресу, после трамвай замедлялся и оставались лишь значения, что уменьшились с последнего отсеивания;
2. Найти все указатели процесса, что указывали на этот адрес;
3. Перезагрузить игру, тем самым, обновив всю память и поменяв адресы. Выполнить первый пункт и среди множества всех указателей с прошлого пункта оставить только те, которые сейчас указывают на новый адрес значения скорости;
4. Выполнять третий пункт до тех пор, пока не будет уверенности в хотя бы одном указателе, который при любых запусках игры всегда указывает на адрес, в котором лежит значение скорости.

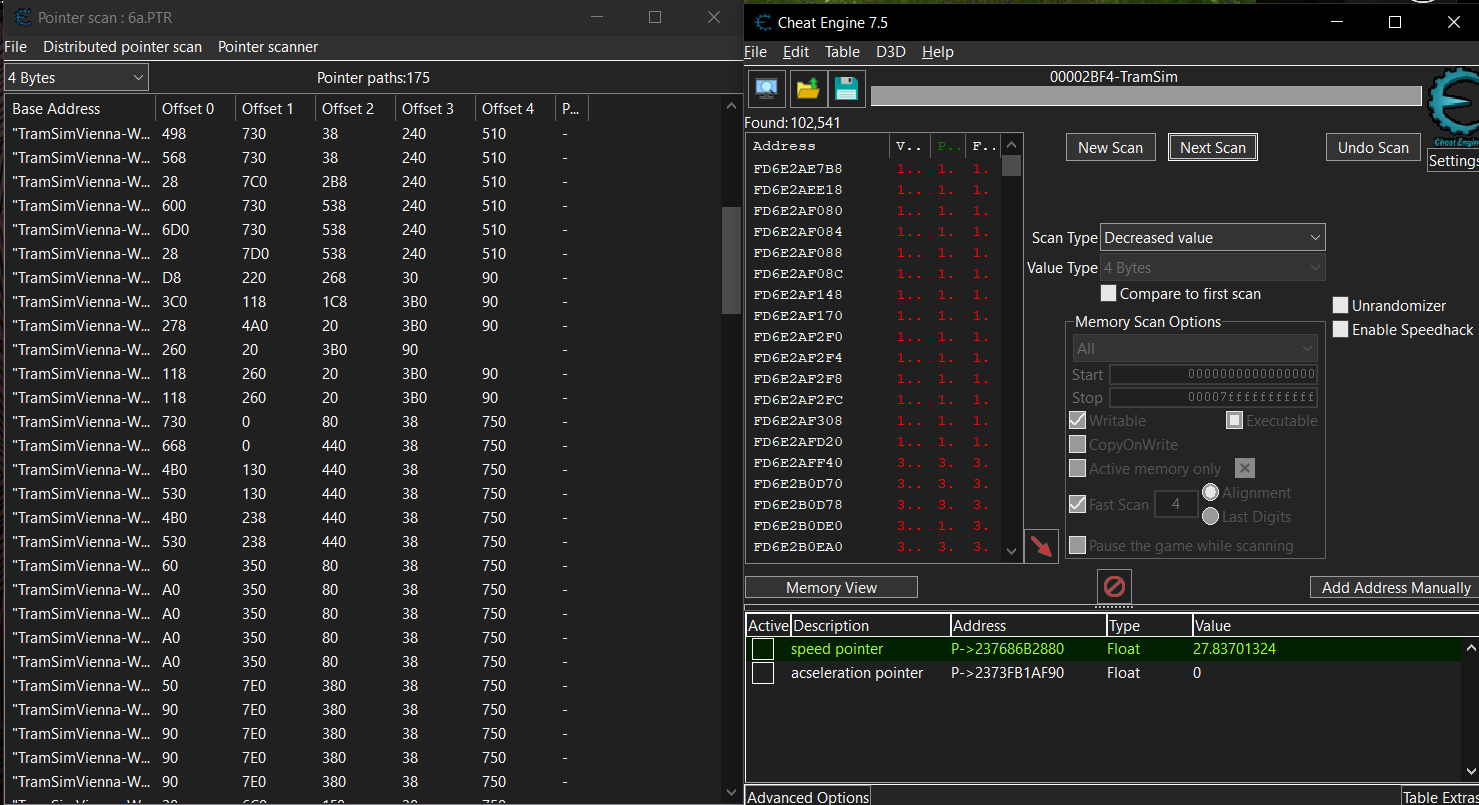


Рисунок 4 – Пример работы Cheat Engine.

Выделив нужный указатель, записываем его данные в программу автопилота, которая, буквально, получает доступ к памяти процесса и читает хранящееся по адресу значение. Эта операция происходит моментально, так как это самая стандартная операция чтения памяти. Благодаря этому автопилот, подобно физической своей версии, способен прямо из процесса работы трамвая получать его скорость и дальше использовать при принятии решений.

Аналогично скорости, было настроено получение из симулятора ускорения трамвая, для более лёгкого его управления.

## **3.3 Программное управление трамваем**

Управление транспортным средством в TramSim Vienna осуществляется при помощи клавиатуры и мыши. Так что для передачи сигналов управления трамваю достаточно эмулировать их, с чем может справиться любой язык программирования. Так как симулятор достаточно отзывчив на короткие сигналы управления в виду того, что игрок управляет ускорением двигателя, было решено не использовать параллельный процесс для управления, а просто занимать некоторое время основного на нажатие клавиши.

# **4 Настройка нейронной сети для решения задачи детекции**

## **4.1 Описание объектов, необходимых для детекции**

Один из упомянутых плюсов писать в симуляции – все возможные ситуации на дороге возможно описать. Переход людей через пешехода, движение впередистоящего автомобиля, светофоры и подобные вещи – то немногое, с чем придётся столкнуться модулю принятия решений. Никаких животных, вышедших на рельсы или неожиданной аварии на дорожных путях. Конечно, в планах писать автопилот с расчётом на то, что он может использоваться и на реальном транспорте, но для обработки таких «форс мажоров» требуется конкретные примеры как изображений таких ситуаций, так и действий водителя, что невозможно получить, используя симуляцию.

Главная задача трамвая – безопасно пройтись по маршруту, останавливаясь на каждой остановке согласно расписанию. Свойство безопасности достигается, по большей части, благодаря следованию сигналам светофоров и резким остановкам в случае, если трамвай ожидает столкновение с каким-либо объектом. То есть сразу для реализации этого свойства нам необходимо считывать с изображения людей, машин и трамвайных светофоров.

Для того, чтобы пройтись по остановкам, необходимо эти самые остановки воспринимать. Для реальных трамваем используются GPS датчики и другие решения от постоянного маршрута. В игре же только при помощи камеры придётся определять места, где трамвай должен остановиться. Благо, это вполне возможно благодаря жёлтым кружкам между рельсами, которые обозначают конкретную точку, где должен трамвай остановиться. Так что точно в объекты детекции необходимо добавить маркеры остановки. Пример изображения такого маркера можно увидеть на рисунке 5.



Рисунок 5 – Маркер остановки.

Таким образом, с помощью перечисленного набора у автопилота получится передвигаться по городу, не нарушая правила дорожного движения, определять трамвайные остановки, избегать случайных столкновений с людьми или машинами, что вполне будет достаточно для автономности.

## **4.2 Обоснование выбора YOLOv8**

Детекция объектов на изображении — это процесс обнаружения и классификации различных объектов или областей интереса на изображении. Существует несколько методов детекции объектов, каждый из которых имеет свои преимущества и недостатки в зависимости от конкретной задачи и условий. Для детекции объектов на видеопотоке наиболее подходят методы, обеспечивающие высокую скорость обработки и точность результатов. В зависимости от конкретных требований и условий видеопотока, можно использовать следующие методы:

1. Методы на основе свёрточных нейронных сетей (CNN): Свёрточные нейронные сети, такие как YOLO (You Only Look Once) и SSD (Single Shot MultiBox Detector), обладают высокой скоростью обработки и способны обнаруживать объекты в реальном времени на видеопотоке. Они являются хорошим выбором для задач, где требуется быстрая детекция объектов.
2. Методы на основе глубокого обучения с использованием регионов (R-CNN, Fast R-CNN, Faster R-CNN): Эти методы также предлагают высокую точность и скорость обработки, основанную на обработке регионов изображения. Они подходят для детекции объектов на видеопотоке с высокой степенью детализации.
3. Методы на основе каскадов Хаара и адаптивных каскадов: Методы, основанные на каскадах Хаара, такие как метод Viola-Jones, обеспечивают быструю детекцию объектов на видеопотоке с низкой вычислительной сложностью. Они могут быть полезны в случаях, когда требуется высокая скорость обработки при достаточной точности детекции.
4. Методы на основе дескрипторов объектов: Локальные дескрипторы объектов, такие как SIFT, SURF, ORB, могут использоваться для детекции объектов на видеопотоке, особенно в случаях, когда объекты имеют высокий уровень детализации или изменчивую текстуру.

Задача детекции достаточно изученная и популярная задача, от того существует множество различных методов. Наша задача состоит в детекции целого ряда объектов на видеопотоке в режиме реального времени. Наиболее предпочтительным для решения этой задачи являются свёрточные нейросети по нескольким причинам:

1. Высокая скорость обработки: Современные архитектуры свёрточных нейронных сетей, такие как YOLO (You Only Look Once) и SSD (Single Shot MultiBox Detector), оптимизированы для быстрой обработки видеопотоков. Они способны обнаруживать объекты в реальном времени с высокой скоростью, что особенно важно для приложений, требующих непрерывного мониторинга.
2. Высокая точность детекции: свёрточные нейросети демонстрируют высокую точность детекции объектов благодаря своей способности извлекать и анализировать признаки изображения на разных уровнях абстракции. Это позволяет им эффективно обнаруживать объекты различных размеров и форм в разнообразных условиях освещения и фона.
3. Способность к обучению на больших наборах данных: свёрточные нейросети позволяют обучать модели на больших объемах размеченных данных, что способствует их адаптации к разнообразным сценариям и условиям работы.
4. Возможность интеграции с графическими процессорами (GPU): Использование графических процессоров для параллельной обработки данных позволяет значительно ускорить вычисления, что делает их идеальным выбором для решения задач детекции на видеопотоке в реальном времени.
5. Гибкость и масштабируемость: Архитектуры свёрточные нейросети можно адаптировать под конкретные требования и условия задачи, а также масштабировать для работы с видеопотоками разного разрешения и качества.

Исходя из этих преимуществ, методы на основе свёрточных нейронных сетей являются оптимальным выбором для решения задачи детекции на видеопотоке в реальном времени, обеспечивая высокую скорость, точность и гибкость работы.

Выбор же именно модели свёрточной нейросети YOLO, а не SSD обусловлен более простотой архитектурой для первого знакомства с нейросетями, а также приятным и активным сообществом, сформированным вокруг архитектуры, что значительно облегчает обучение и решение самых разных проблем. Кроме того, YOLO хорошо справляется с детекцией объектов различных размеров на изображении, включая как крупные, так и мелкие объекты, благодаря механизму деления изображения на сетку и применению множества анкоров, что позволяет начинать строить картину окружающего мира задолго до необходимости взаимодействовать с объектами. Пример работы YOLO можно увидеть на рисунке 6



Рисунок 6 – Пример работы YOLO.

В 2024 году наиболее актуальной версией YOLO считается YOLOv8. Она обладает следующими преимуществом в виде лёгкой архитектуры, которая фокусируется на скорости и эффективности, а также имеет новые функции, такие как пользовательские привязки и трансферное обучение, которые делают модель проще в обучении и настройке для решения конкретных задач.

Также стоит сказать, что YOLOv9 и YOLOv6 также являются актуальными в наше время. Однако YOLOv9 хоть и превосходит YOLOv8 по точности, но имеет более медленное время вывода, а YOLOv6 лучше работает с небольшими объектами, но имеет более высокую вычислительную задержку, что не подходит для нашей конкретной задачи детекции.

## **4.3 Подготовка датасета и обучение модели**

Так как камера во время движения автопилота будет находится на носу трамвая, то и требуемый для обучения нейросети датасет тоже будет формироваться из записей с камеры в том же положении. Пример кадра с камеры можно увидеть на рисунке 7. Так как симулятор TramSim является обычной видеоигрой, то запись датасета происходила достаточно легко – с помощью наигранных часов и записи экрана. В дальнейшем из видеозаписи игрового процесса выбирались кадры, которые были достаточно богаты различными игровыми объектами.



Рисунок 7 – Вид камеры на носу.

В итоге из, суммарно, около часа записей игрового процесса было получено 109 различных кадров, готовых к разметке. Так как для решения задачи детекции использовалась модель YOLO, то для разметки было достаточно выделить на кадрах искомые объекты в прямоугольники, подписав каждый из них. Для удобства выполнения этого монотонного процесса было решено воспользоваться интернет-сервисом Roboflow.

Roboflow — это платформа, предназначенная для упрощения процесса подготовки данных для обучения моделей машинного обучения. Она предлагает различные инструменты и функции, которые помогают исследователям данных, разработчикам и инженерам в создании и управлении датасетами. Roboflow достаточно большой сервис, он обладает следующими возможностями:

1. Преобразование данных: позволяет быстро преобразовывать и аугментировать изображения и другие данные для обучения моделей, включая изменение размеров, повороты, изменение контрастности и многое другое.
2. Аннотирование данных: предоставляет средства для разметки изображений и других типов данных, необходимых для обучения моделей, таких как метки объектов на изображениях.
3. Интеграция с различными фреймворками: поддерживает интеграцию с популярными фреймворками машинного обучения, такими как TensorFlow, PyTorch, Keras и другими.
4. Управление датасетами: позволяет организовывать и управлять датасетами, включая загрузку, хранение и обмен данными.
5. Обучение моделей: предоставляет возможность обучения моделей машинного обучения на подготовленных данных.
6. Коллаборация: предоставляет средства для совместной работы над проектами и обмена данными и результатами между членами команды.

Roboflow помогает упростить и автоматизировать рутинные процессы, связанные с подготовкой данных для машинного обучения, что позволяет исследователям и разработчикам сосредоточиться на создании и улучшении моделей. Благодаря возможностям подключения дополнительных людей в команду разметки и обучения моделей прямо в их сервисе, не было страха того, что исследования могут уткнуться в потолок и занять очень много времени. К счастью, опасения не подтвердились и Roboflow был использован только как сервис для хранения датасета и последующего аннотирования.

Кроме того, прямо в сервисе имеющийся датасет можно было привести в форму, необходимую для обучения YOLO, а также применить аугментирование, расширив имеющийся датасет до 257 фотографий. Изображение датасета можно увидеть на восьмом рисунке. Там же датасет был разделён на тренировочный, валидирующий и тестирующий наборы в соотношении 86 к 10 к 4. Аугментирование позволило получить достаточно большой датасет для обучения, которого вполне хватило для обучения автопилота, а разделения данных на наборы улучшило результаты.

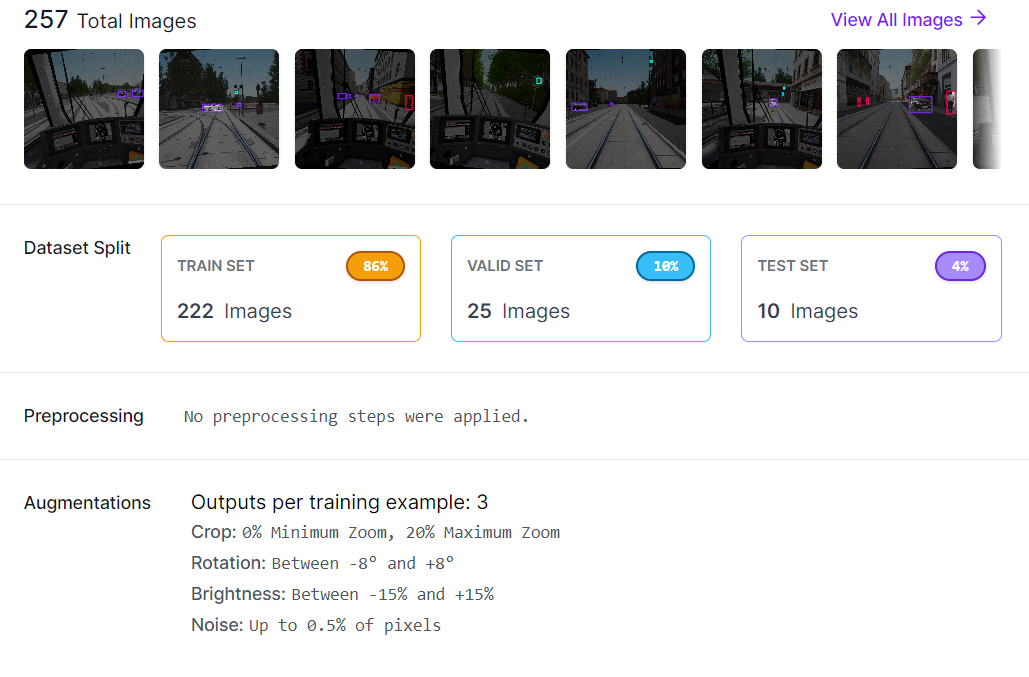


Рисунок 8 – Датасет на странице Roboflow.

В итоге, имеющийся датасет был использован для дообучения yolo8n, изначально обученной на стандартном датасете coco128. Версия nano была выбрана в виду того, что от модели требовалось моментально реагировать на происходящие события на дороге трамвая, точность определения объекта была на втором месте по важности.

Теперь требовалось обучить нейросеть, для этого использовался собственный ноутбук с видеокартой RTX 3050 Ti Laptop. Облачные сервисы не применялись в виду долгой настройки и вполне подходящего под задачу оборудования. В результате экспериментов по обучению сети, лучшей версией, согласно метрикам precision, recall и mAP50, была выбрана модель, обученная на 60 эпохах на имеющемся датасете. Именно это количество эпох оптимально для того, чтобы сеть максимально обучилась, но не имела проблем на валидирующем и тестировочном наборах. Результаты обучения представлены на рисунке 9.

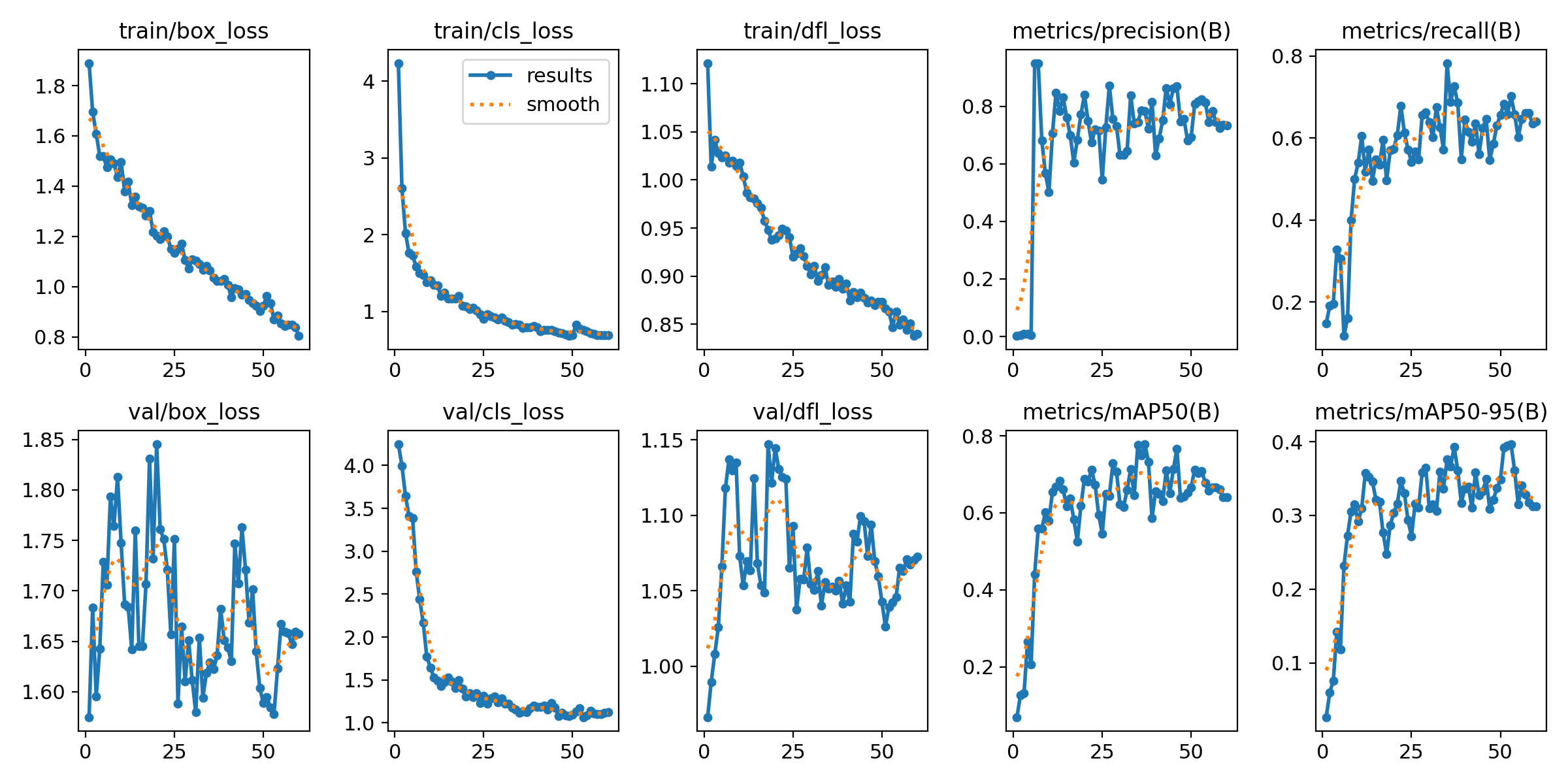


Рисунок 9 – Результаты обучения YOLOv8 на 60-ти эпохах.

# **5 Разработка модуля принятия решения**

## **5.1 Обработка размеченного кадра из YOLO**

В прошлых главах были перечислены объекты, которые должна находить нейронная сеть и размечать на кадре. Люди и машины, чтобы избегать столкновений, светофоры, чтобы соблюдать правила, и жёлтые кружки, чтобы знать где останавливаться. Теперь же автопилот должен, исходя из ситуации вокруг него, определять своё дальнейшее поведение. Однако передавать в модуль принятия решений просто кадр с координатами размеченных прямоугольников недостаточно, необходимо обработать его так, чтобы с ним можно было удобно работать.

Трамвай не должен сталкиваться с пешеходами и автомобилями. Для этого, он должен понимать расстояние до них и находятся ли они по направлению движения транспорта. То есть нам нужно получить координаты объекта в полярной системе. Так, учитывая скорость, автопилот будет принимать решение, нужно ли ему начать резкую остановку. Для того, чтобы из прямоугольника объекта на изображении получить расстояние и угол, необходимо произвести вычисления.

Чтобы вычислить расстояние, нужно вычислить нынешнюю высоту объекта и разделить максимальную высоту объекта этого типа на нынешнюю. Так как в симуляторе используются ограниченное количество моделей прохожих и машин, то возможно вручную задать максимальную высоту объекта, которая будет равна высоте объекта в кадре, при которой трамвай должен начать резкую остановку. Можно сказать, максимальная высота – это высота объекта в кадре, когда он находится уже слишком близко к трамваю и есть шанс с ним столкнуться.

С углом более сложно. Для начала в симуляторе необходимо задать определённый угол обзора камеры, например, 90 градусов. Изображение, которое видит камера, плоское, однако угол до объекта мы должны оценивать по дуге окружности. По сути дела, чем дальше от правого края изображения находится объект, тем больше его угол. Однако изменение величины угла от расстояния имеет не линейную зависимость. С помощью стандартной тригонометрии можно вычислить с помощью формулы (1).

где

– угол обзора камеры,

– ширина кадра в пикселях,

– на сколько сдвинута по ширине середина рамки объекта.

В результате получаем значение угла в радианах от 0 и до , где угол – это угол по направлению движения трамвая.

Теперь, вычислив расстояние и угол объекта, можно удобно с ним работать в модуле принятия решений. Для простоты восприятия в дальнейшем, было решено перевести полярные координаты в декартовы.

Касательно же светофоров – они не находятся на пути следования трамвая, а значит автопилоту не нужны их координаты, а только сигнал, который они показывают. Разрешающий сигнал выглядит как две вертикальные или две диагональные точки, а запрещающий как три горизонтальные. Чтобы определить, какой сигнал показывает светофор, необходимо обработать его изображение. Так как точки белые на чёрном фоне, то с помощью фильтра цвета их достаточно легко отделить от фона. Лишние шумы можно убрать с помощью эрозии с последующим расширением, а вернуть количество пикселей с помощью повторного расширения. В итоге получаем изображение, в котором находим самый верхний левый и самый нижний правый не чёрные пиксели. Они определяют границы сигнала, который показывает светофор. Если рамка получилась продолговатой, то считаем, что знак запрещающий, если же вытянутой или квадратной, то знак разрешающий. Пусть этот метод не даёт абсолютной точности, но он быстро реализуем и его хватает для решаемой задачи.

Для жёлтого круга остановки достаточно просто определить, как далеко от конца кадра в процентах он находится. Этого относительного расстояния будет достаточно, чтобы автопилот понял, что, как только круг пропадёт из виду, настанет время полностью останавливаться.

В результате, все объекты теперь представляют собой не подписанные рамки, а только полезный набор данных, который можно сложить в одну единую модель мира, с которой и будет работать модуль принятия решений.

## **5.2 Выбор алгоритма принятия решения**

Теперь все части для получения полной информации о мире вокруг трамвая готовы. Автопилот способен считывать значения скорости, а ускорения трамвая, а также, благодаря нейросети, определять людей, машины, светофоры и кружки для остановки. Имея всю эту информацию, автопилот вполне способен принимать нужные решения наравне с игроком в симулятор. Но для этого необходимо реализовать модуль принятия решений.

Модули принятия решений для автопилотов могут быть реализованы с использованием различных подходов и технологий. Некоторые из наиболее распространенных способов включают:

1. Правила и эвристики: подход основан на заранее заданных правилах и эвристиках, которые определяют оптимальные действия в различных ситуациях. Например, автопилот может следовать набору правил для выполнения маневров, таких как взлет, крейсерский полет и посадка.
2. Логическое программирование: метод использует логические выражения и правила для описания принятия решений. Модуль принятия решений анализирует текущую ситуацию и применяет логические правила для выбора оптимального действия. Подобные автопилоты писались на заре зарождения логического программирования для написания экспертных систем.
3. Нейросетевые модели: подход включает использование искусственных нейронных сетей для обучения модуля принятия решений на основе больших объемов данных. Нейросеть может обучаться на исторических данных о заездах и аварийных ситуациях, чтобы прогнозировать оптимальные действия.
4. Гибридные подходы: комбинация нескольких методов, например, комбинирование правил и эвристик с алгоритмами машинного обучения для достижения оптимальных результатов.

Каждый из этих подходов имеет свои преимущества и ограничения, и выбор конкретного метода зависит от требований к автопилоту. Безусловно, эффективнее всего было бы использовать нейросеть, обученную на множестве размеченных записей управления трамваем. Однако этот метод требует очень больших усилий для создания такого датасета, подготовки правильной структуры нейросети и подготовки необходимого для обучения оборудования, что по силу только большим командам разработки. И даже так это займёт достаточно много времени.

Поэтому было решено воспользоваться способами задания механического поведения при помощи хорошо знакомого конечного автомата, в котором каждое из состояний соответствует некоторому поведению.

## **5.3 Описание конечного автомата состояний**

Каждую итерацию автопилот будет формировать модель мира и считывать данные о скорости, после чего на основе этих данных и результатов прошлых итераций будет приниматься решение либо о переходе в новое состояние, либо продолжение выполнение поведения предыдущего. Получив итоговое состояние, автопилот посылать некоторые сигналы управления трамваю, соответствующее состоянию, после чего начнёт новую итерацию. Сами сигналы управления, как упоминалось, должны быть короткими, чтобы автопилот мог быстро и детально реагировать на изменения в мире вокруг.

Были выделены следующие состояния и реализации их поведения:

1. Движение – трамвай едет вперёд, придерживаясь небольшого интервала скорости. Во время этого состояния трамвай в любой момент готов начать торможение;
2. Быстрое движение – когда вокруг нет объектов, о которых стоит переживать, трамвай начинает двигаться быстро;
3. Остановка – трамвай заметил впереди жёлтый кружок и медленно начинает подъезжать к точке остановки;
4. Посадка-высадка – трамвай прибыл на остановку. Он полностью останавливается, открывает двери, дожидается посадки и после вновь их закрывает. В отличие от других состояний, посадка-высадка реализована как скрипт, выполняемый за определённое время.
5. Ожидание светофора – трамвай увидел запрещающий сигнал светофора и ждёт разрешающий. Трамвай начинает замедляться, чтобы не проехать светофор.
6. Резкая остановка – трамвай заметил опасность на своём пути и вынужден быстро остановиться, чтобы избежать аварии.

На данный момент этого набора вполне достаточно, чтобы выполнять все необходимые функции общественного транспорта. Возможно, для большей точности действий в последующих версиях будут добавлены ещё несколько моделей поведения или больше эвристических функций для правильности переходов между состояниями.

Большую часть времени автопилот будет находить в состоянии движения, после чего из него переходит в состояния особого поведения, из которых он вновь будет возвращаться в движение. В каждом из состояний, кроме основного, автопилот не может слишком долгое время, если на то нет особой причины, что гарантирует то, что транспорт нигде не застрянет по пути следования. Графическое изображения конечного автомата с условиями перехода можно рассмотреть на рисунке 10.

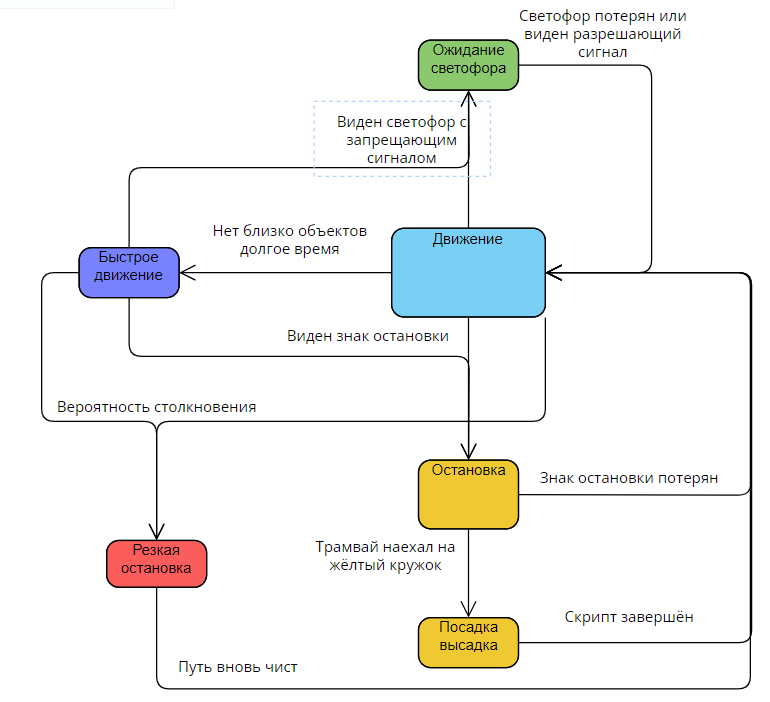


Рисунок 10 – Конечный автомат состояний автопилота.

# **6 Итоговая схема работы автопилота**

Одним из наиболее популярных языков программирования для анализа данных является Python. Этот язык стал выбором многих специалистов благодаря обширному набору библиотек, которые позволяют решать практически любые задачи, связанные с анализом данных, машинным обучением и искусственным интеллектом.

Альтернативой Python мог бы стать язык программирования C++, который известен своей высокой производительностью. Программы, написанные на C++, зачастую работают быстрее и эффективнее, так как язык предоставляет больше возможностей для оптимизации кода. Однако, программирование на C++ требует более глубоких знаний и умений, и процесс разработки на этом языке занимает больше времени.

В данной работе не стояла цель создать максимально производительный автопилот. Основное внимание уделялось простоте и скорости разработки, что сделало Python наиболее подходящим выбором. Использование Python позволяет быстро прототипировать и тестировать идеи, что особенно важно на этапе исследования и разработки.

Все реализованные модули были разделены на соответствующие файлы:

1. speed\_from\_memory.py – содержит функции обращения к памяти процесса «TramSimVienna-Win64-Shipping.exe», а именно имеет возможно читать значения скорости и ускорения для записанных в файл указателей;
2. output.py – модуль работы с потоком выходных команд;
3. tramsim\_windows\_capture.py – содержит все необходимые объекты для библиотеки windows\_capture. При инициализации модуля создаёт отдельный поток, который постоянно считывает кадры игры и записывает их в глобальную переменну. Для того, чтобы добиться синхронизации потоков было решено воспользоваться примитивными блокировками потока.
4. yolo\_predict.py – модуль нейросети, работающий с моделью yolo и кадром игры. Производит с помощью yolo детекцию объектов на изображении, после чего обрабатывает их, собирая данные в модель мира, которую и возвращает следующему модулю.
5. decision\_module.py – модуль принятия решений. Получает из предыдущего модуля модель мира и скорость, чтобы на основании новых и прошлых данных сделать выбор о переходе автопилота в иное состояние, если это требуется. Также в этом модуле описаны сигналы управления, которые автопилот должен передать трамваю в каждом из состояний.
6. main.py – запускаемый файл автопилота, в котором соединены все модули. Инициализирует модули, а после использует их функции в нужной последовательности внутри цикла, поддерживая работу автопилота.

Такая модульная система позволяла чётко структурировать программу, разделив её на отдельные компоненты, каждый из которых выполнял определённые функции. Это обеспечивало гибкость и удобство в разработке и поддержке кода. В случае необходимости замену реализации одного модуля можно было выполнить без изменения других модулей, благодаря чётко определённым интерфейсам и контрактам между ними.

Эта архитектура, реализуемая с самого начала проекта, неоднократно оправдывала свои преимущества. Например, когда возникала необходимость обновления или замены устаревших модулей, такие изменения можно было внести быстро и безболезненно, не затрагивая остальную часть системы. Это значительно упрощало процесс разработки и тестирования, позволяя команде сосредоточиться на улучшении конкретных частей программы, не опасаясь нарушить её общую работу. Полный набор файлов проекта можно увидеть на рисунке 12.

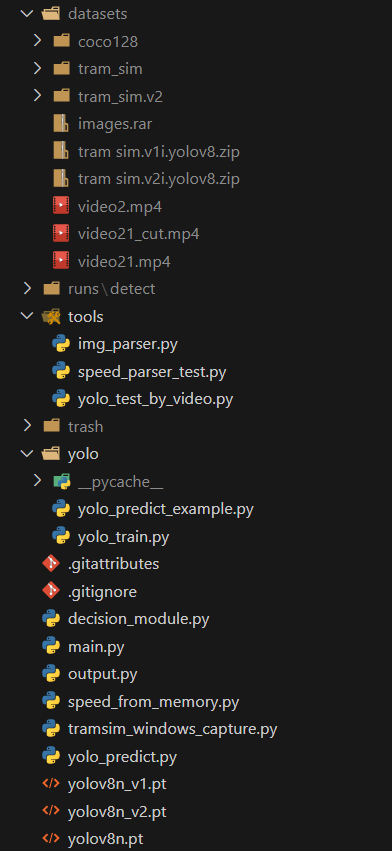


Рисунок 12 – Файловая система проекта автопилота.

В итоге, созданная программа, состоящая из нескольких файлов, запускается и начинает получать данные из памяти процесса и захвата экрана. Данные из памяти процесса включают параметры, такие как ускорение и скорость трамвая. Одновременно программа захватывает изображения с экрана, чтобы анализировать визуальную информацию из симулятора.

Обработав полученные данные, программа генерирует команды управления для симулятора трамвая. Эти команды регулируют скорость, направление движения, а также остановку и запуск трамвая. Таким образом, программа автоматически управляет трамваем, используя комбинацию данных для принятия решений в реальном времени.

# **7 Тестирование системы и результаты**

Тестирование работоспособности модулей системы проходило на протяжении всего процесса написания автопилота. Однако, полноценное оценивание работоспособности автопилота проводилось только тогда, когда каждый из модулей был работоспособен.

Во время тестирования сильно менялся модуль принятия решения, общие правила которого обрастали деталями и точностями. Кроме того, в модуле использовались некоторые константные значения, например разрешённой скорости, которые необходимо было выяснить опытным путём. Подобно обучению нейронной сети, если бы её было решено использовать в модуле принятия решений, все эвристики и логические правила настраивались вручную после каждого прохода по маршруту.

Самым проблематичным модулем оказался модуль детекции. Периодические ложные срабатывания для знаков остановки или светофоров вносили неконтролируемое поведение в алгоритм. Благо, большую часть проблем удалось обработать при помощи логических выражений внутри модуля принятия решений и во время обработки модели мира. Для того, чтобы этих проблем и вовсе не возникало, точно необходимо было увеличить датасет, дополнив его кадрами, на которых было ложное срабатывание. Модули управления и получения информации работали безотказно на протяжении всех тестов. Модуль принятия решений из-за того, что собрал в себе всего 6 состояний, возможно было полностью просмотреть на наличие ошибок, что и было сделано.

В результате, полученный автопилот отлично справляется с передвижением по городу без аварийных ситуаций и посадкой-высадкой пассажиров. Подавляюще большинство светофоров автопилот определяет правильно и соблюдает правила дорожного движения, но для некоторых необычных светофоров, например, двойных, он срабатывает ложно. Впрочем, за время тестов все ложные срабатывания не привели к столкновению, хотя правила были явно нарушены. Также есть замечание касательно скорости передвижения трамвая. Она недостаточно быстрая в виду того, что детекцию светофоров и кружков остановки не удаётся проделать на достаточном расстоянии, чтобы трамвай смог успеть остановиться перед ними без аварийного тормоза. Эту и прошлую проблему ложных срабатываний вполне можно решить увеличением датасета и лучшим обучением нейронной сети, ибо все остальные модули уже вполне готовы к более детальной обработке мира.

Также было замечено, что для ещё более точного построения модели мира было бы правильно не только обрабатывать нынешнюю модель и предыдущие, но и ещё и создавать предсказания модели мира, основываясь на предыдущем опыте. Это вполне математическая задача назначения множеству объектов векторов их скорости и последующая коррекция положения и скорости объектов, при получении реально картины миры. Но это было решено опустить в рамках данной работы в виду того, что поведение пешеходов и машин внутри симуляции вполне предсказуемо и оно не может выйти за вполне обрабатываемые рамки, что, конечно, для реального мира будет слишком весомо.

# **ЗАКЛЮЧЕНИЕ**

В ходе выполнения данной работы были реализованы все необходимые модули необходимые для полной работоспособности автопилота для симулятора трамвая. При помощи Roboflow был создан необходимый для обучения нейросети датасет, на собственном устройстве обучена модель yolov8n для решения задачи детекции объектов на изображениях. Задача получения скорости была закрыта при помощи прямого получения значения скорости из памяти процесса, указатель на адрес которой был получен при помощи Cheat Engine. Среди всех рассмотренных вариантов реализации модуля принятия решения был выбран пусть и не самый эффективный, но вполне реализуемый метод логического программирования и задания правил и эвристик. На основе этих алгоритмов была разработана машина состояний, определяющая поведения трамвая при каждой итерации. Детекция объектов, создание модели мира, получение скорости и ускорения, принятие решения о состоянии трамвая и соответствующий состоянию вывод сигналов управления – полный цикл работы разработан и на тестах показал свою отличную работоспособность.

Были намечены возможные пути улучшения работы автопилота. Главной задачей для стабильности работы алгоритма была выделена задача об увеличении датасета для обучения нейронной сети, в котором особенное внимание стоит уделить распознаванию объектов из далека, а также некоторым объектам, на которых происходило ложное срабатывание детекции. Другой важной задачей с практической точки зрения можно выделить создание модуля предсказания движения объектов, хотя, как было ранее указано, без него автопилот внутри симулятора вполне успешно справляется.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1 Статья о результатах работы Cognitive Pilot в Питере – Текст: электронный // Фонтанка.ру: официальный сайт. – URL: https://www.fontanka.ru/2024/03/14/73330787/ (дата обращения: 11.05.2023).

2 TramSim Vienna – Текст: электронный // TramSim, ViewApp: официальный сайт. – URL: https://www.tram-sim.com/en/vienna (дата обращения: 03.12.2023).

3 Статья от Cognitive Pilot «Как наш беспилотный трамвай видит реальный город» – Текст: электронный // Хабр: официальный сайт. – URL: https://habr.com/ru/companies/cognitivepilot/articles/498660/ (дата обращения: 05.01.2024).

4 Обнаружение объектов YOLOv8 – Текст: электронный // Ultralytics: официальный сайт. – URL: https://docs.ultralytics.com/ru/tasks/detect/ (дата обращения: 05.01.2024).

5 Обучение по Cheat Engine – Текст: электронный // Cheat Engine: официальный сайт. – URL: https://www.cheatengine.org/tutorials.php (дата обращения: 05.01.2024).

6 Николенко, С. И. Глубокое обучение : учебник / С. И. Николенко, А. А. Кадурин, Е. В. Архангельская – Санкт-Петербург : Питер, 2018. – 480 с.; 23 см. – ISBN 978-5-496-02536-2. – Текст : непосредственный.