

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ
Факультет Кафедра
машиностроения и транспорта «Транспортные машины»
(наименование) (наименование)

Направление подготовки

23.03.01 «Технология транспортных
процессов»
(код и наименование)

Профиль

«Организация и безопасность движения»
(наименование)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
на тему
«Разработка информационной системы парковки
беспилотного средства в условиях города»

Студент

(подпись, дата)

Евсеева Ольга Сергеевна
(фио полностью)

Научный руководитель
к.т.н., доцент

(подпись, дата)

Генералова А.А.
(фамилия, инициалы)

Нормоконтролёр
к.т.н., доцент

(подпись, дата)

Логинов О.Н.
(фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от _____ № _____)

Заведующий кафедрой

(подпись, дата)

Салмин В.В.
(фамилия, инициалы)

Работа защищена с отметкой _____ (протокол заседания ГЭК
от _____ № _____)

Секретарь ГЭК
к.т.н., доцент

(подпись, дата)

Метальникова О.К.
(фамилия, инициалы)

Пенза 2017

ФГБОУ ВО «ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ
ФАКУЛЬТЕТ МАШИНОСТРОЕНИЯ И ТРАНСПОРТА
КАФЕДРА «ТРАНСПОРТНЫЕ МАШИНЫ»

«Утверждаю»

Заведующий кафедрой ТМ
д.т.н., проф. _____ В.В. Салмин
подпись

« ____ » _____ 2016 г.

ЗАДАНИЕ
на выпускную квалификационную работу

Студенту гр. 13МГ1 Политехнического института, очной формы обучения по направлению 23.03.01 «Технология транспортных процессов»

Евсеевой Ольге Сергеевне

Научный руководитель работы: Кандидат технических наук, доцент
А.А. Генералова

Время выполнения дипломного проекта: с 10.12.2016 г. по 08.06.2017 г.

Место преддипломной практики: Управление транспорта и связи г. Пензы

Тема работы: «Разработка информационной системы парковки беспилотного средства в условиях города»

Тема утверждена приказом ректора ПГУ № 1185/С от 02 декабря 2016 г.

Техническое задание на работу (назначение устройства, условия применения, внешние воздействия, специальные требования и т.п.): разработать алгоритмы управления автономными транспортными средствами.

Произвести экономическую оценку затрат на реорганизацию проекта беспилотного транспортного средства.

Объём и содержание основной части работы

Пояснительная записка (перечень вопросов, подлежащих разработке, расчетов, обоснований, описаний):

- введение с обоснованием актуальности темы исследования, постановкой цели и задач исследования – 1 – 2 страницы;
- теоретический анализ проблемы беспилотного транспорта во взаимосвязи с поставленной целью и задачами, 20 – 30 страниц;
- организационно-управленческий раздел - разработка модулей работы с векторами, отрисовки векторов, сенсоры, отрисовки сенсоров, обнаружения коллизий, формирования поля коллизий, модуля перемещений и принятия решений - 10 – 30 страниц;
- экономический раздел – обоснование реальных затрат на реализацию предложенного проекта 4 – 10 страниц;

- заключение – выводы и обоснования по реализации разработанного комплекса мероприятий – 1-2 страницы.

Графическая часть (перечень и содержание чертежей, плакатов)

1 лист «Цели и задачи выпускной работы» А4;
2 лист «Анализ существующих беспилотных транспортных средств» А4;
3 лист «Система автоматической и автономной парковки автомобиля» А4;
4 лист «Разработанные библиотеки функций» А4;
5-10 листы «Описание библиотек» А4;
11 лист «Алгоритм работы одной итерации жизни объекта-пилота» А4;
12-13 листы «Обнаружение столкновений» А4;
10 лист «График изменения скорости» А4;
11 лист «Экономическое обоснование предложенных мероприятий» А4;
12 лист «Основные выводы и рекомендации» А4.

Рекомендуемая литература:

1 Аизерман М.А. Автоматика переключения передач[Текст].- М.: Машгиз, 1948.-140 с.

2 Балабин, И.В Испытания автомобилей /Балабин И.В, Куров Б.А, Лаптев С.А. / 1998. - 192 с.[Электронный ресурс]. Режим доступа:http://www.belsis.ru/news/popularnaja_elektronika/volvo-self-parking.html

3 Гришкевич, А.И Автомобили-роботы испытания[Текст] /Гришкевич А.И, Высоцкий М.С./. 1991.-187с.

4 Дрю Д. Теория транспортных потоков и управление ими. М.: Транспорт, 1972.[Электронный ресурс]. Режим доступа:<http://howcarworks.ru>

Дата выдачи задания: 10.12.2016 г.

Руководитель дипломного проекта:

ст. преподаватель каф. «ТМ» _____ А.А. Генералова

Задание к исполнению принял студент

гр. 13МГ1 _____ О.С. Евсеева

10 декабря 2016 г.

РЕФЕРАТ

Евсеев О.С. Разработка информационной системы парковки беспилотного средства в условиях города / Ольга Сергеевна Евсеева– Пенза: ПГУ, 2017. – 82 страницы, 3 части, 58 рисунков, 4 таблицы, 30 источников, 2 приложения.

БЕСПИЛОТНЫЙ АВТОМОБИЛЬ, БЕСПИЛОТНОЕ ТРАНСПОРТНОЕ СРЕДСТВО, АВТОМОБИЛЬ РОБОТ, ВИДЕОКАМЕРА, РАДАР, ЛАЗЕР, ПОЛЕ КОЛЛИЗИИ, ПАРКОВКА, ПРОЦЕСС МОДЕЛИРОВАНИЯ, ОБЪЕКТ-ПИЛОТ.

Объект исследования - беспилотный автомобиль.

Предмет – информационная система парковки беспилотного транспортного средства.

Цель работы: разработка бортовой информационно-измерительной и управляющей системы автопилота серийно выпускаемого автомобиля для парковки в условиях города.

Выпускная квалификационная работа состоит из введения, трёх разделов, заключения, списка литературы и приложения.

Первый раздел посвящен изучению существующих автономных транспортных средств и способу их работы. Были изучены системы автоматической и автономной парковки автомобиля. Произведен анализ проблем возникающих у автопилота.

Во втором разделе была произведена разработка алгоритмов управления автономными транспортными средствами. Разработан вариант определения нахождения автомобиля в текущий момент времени. Произведен расчет коллизии. Описаны библиотеки, используемые в работе.

В третьем разделе были рассчитаны затраты на проектирование разработки автономной парковки. Определяется полный перечень работ, трудоёмкость работ. Распределяются работы между исполнителями. Определяются длительность цикла выполнения этапа и затраты на проектирование, т.е. составляется смета.

Содержание

Введение.....	4
1 Анализ современного состояния беспилотного автомобиля.....	7
1.1 Автономные транспортные средства и автомобили-роботы	7
1.2 Система автоматической и автономной парковки автомобиля	14
1.3 Проблемы автопилота.....	18
Выводы	23
2 Разработка алгоритмов управления автономными транспортными средствами.....	24
2.1 Система навигации.....	24
2.2 Обнаружение столкновений.....	27
2.3 Описание библиотек	41
Вывод.....	64
3 Затраты на реализацию проекта	65
3.1 Расчет предпроизводственных затрат.....	65
Вывод.....	76
Заключение	77
Литература	Ошибка! Закладка не определена.
Приложение А (обязательное) Презентация бакалаврской работы.....	82
Приложение Б (информационное) Листинг программного кода.....	99

Введение

На сегодняшний день автомобиль преимущественно является одним из наиболее эффективных, доступных и востребованных наземных транспортных средств. «Вследствие этого, одной из ярко выраженных мировых тенденций последних десятилетий является автомобилизация, то есть использования автомобильного транспорта в различных сферах человеческой деятельности увеличивается. Автоматизация производственного процесса, строгое тестирование новых моделей и версий автомобиля обеспечивает безопасность и удобство для людей. Потребность в увеличении выпуска и улучшения структуры автомобилей, более полно отвечающих потребностям народного хозяйства, требуют экономии топлива, повышения ресурса и снижения металлоемкости»[1]. В связи с данным увеличением экономичности, значительной прочности, динамические и ездовые особенности машины, понижение его массы и снижение вредоносного влияния на находящуюся вокруг сферу, считаются главными вопросами автопромышленности на сегодняшний день. С целью свершения требуемых условий в структуре комплекса автоматизированного испытания необходимо создать электропривод автономной системы управления машиной с целью циклических испытательных заездов. «Задачами проектирования мобильных робототехнических комплексов, их информационно-измерительных и управляющих систем занимались российские и зарубежные ученые: В.П. Андреев, В.Л. Афонин, С.А. Воротников, Е.А. Девянин, М.Б. Игнатьев, П.Д. Крутько, Ю.В. Подураев, Л.Б. Рапопорт, А.М. Формальский, Н. Asada, Y. Kuwata, M. Liu, A. Nüchter, S. Thrun, A. Waj-Fraj , A. West и другие»[2].

С учетом зачастую изменяющихся обстоятельств деятельность водителей требует значительного количества управляющих воздействий и стабильной концентрации интереса, что содействует стремительной утомляемости. В соответствии с основной концепцией, сегодняшние научные

работники показывают на то, что и у грамотных водителей с огромным стажем уже после 4-5 часов постоянного управления автотранспортным средством из-за утомления уменьшается внимание и в 1,5-2 раза увеличивается число погрешностей управления, которые увеличивают возможность появления небольших и средних дорожно-транспортных происшествий.

Опытные примеры разрабатываемых машин новейшего поколения, точно также как и контрольные 5 экземпляры массового издаваемых автотранспортных средств, постоянно подвергаются строгим проверкам в тестовом полигоне, с целью раскрытия путей оптимизации их системы. Данное дает возможность вовремя исполнять возрастающие требования покупателей к приспособленности авто к активной эксплуатации в трудных дорожных обстоятельствах, прочности коммерческих автомобилей, затрате горючего, ремонтпригодности в полевых условиях и многое другое. Деятельность бригады испытателей – сложный, однообразный труд, взаимосвязанный с опасным для самочувствия степенью вибраций и тряски. «Современные цифровые системы автоматического управления (САУ) в состоянии решить практически сколь угодно сложные алгоритмы вне зависимости от принципа действия устройств» [3]. Этим наиболее приоритеты при формировании средств автоматики сместились в сферу информативных инноваций. С иной стороны, вследствие формирования вычислительной техники, присутствие отыскивания решений проблем этого класса начали обширно использоваться способы имитационного математического прогнозирования. «Занимая минимальное время (что немаловажно с учетом сокращения сроков создания новой техники) и требуя меньших расходов, нежели традиционные методики натурных испытаний, моделирование при помощи ЭВМ открывает новые возможности для углубленного изучения характера влияния нового оборудования и алгоритмов управления на происходящие процессы и результирующие эксплуатационно-экономические показатели»[4].

Объект исследования: беспилотный автомобиль.

Предмет исследования: информационная система парковки беспилотного транспортного средства.

Цель выпускной работы - разработка бортовой информационно-измерительной и управляющей системы автоматической парковки автомобиля в условиях города.

Основные задачи исследования:

- 1 Рассмотреть существующие беспилотные транспортные средства;
- 2 Разработать алгоритмы траекторного управления транспортным средством, реализующих маневры с учетом возможности огибания препятствий;
- 3 Реализовать программное обеспечение узлов и блоков бортовой информационно-измерительной и управляющей системы;
- 4 Оценить затраты на реализацию проекта.

1 Анализ современного состояния беспилотного автомобиля

1.1 Автономные транспортные средства и автомобили-роботы

Повышение автономности автомобилей стали рассматривать относительно давно и первые исследования в этом направлении проводились в Японии в 1970 году. Сегодня эта область продолжает процветать, по всему миру идут активные разработки. «Автономные транспортные средства уже проехали многие сотни километров, и создатели данной технологии утверждают, что она поможет снизить количество заторов на дорогах, увеличить пропускную способность дорог и сделать движение более безопасным, без участия водителя, так как по статистике около 90 % всех дорожно-транспортных происшествий происходит по вине человека»[5].

Робототехника – отрасль, которой отводится ключевая роль в большинстве экспертных прогнозов развития ведущих стран мира. «Основным направлением современного динамичного прогрессирования этой области служит не только количественный рост инновационных решений и инвестиций, но и их переориентация от сектора промышленной робототехники, к которой традиционно относят различные программируемые манипуляторы для сварки, сортировки и прочее, в сектор когнитивных роботов, возможности которых уже позволяют выйти за границы естественных для них рынков рекламы и развлечений»[5].

В сфере проектирования беспилотных легковых автомобилей-роботов оптимального преуспевания достигла фирма Google (Соединенных Штатов Америки). Перемещение Google-мобиля контролируется согласно дорожной карте, какую составляет концепция управления. Согласно дорогам Калифорнии машины Google в пробном режиме проехали наиболее 300 тыс. миль, в этом количестве согласно основной магистрали среди Сан-Франциско и Лос-Анджелесом. Значительная доля этой дороги освоена целиком в автоматическом режиме с работником фирмы в пассажирском сиденье.

В автопарке компании Google имеется 20 беспилотных автомобилей моделей Toyota Prius, Audi TT и другие (рисунок 1.1).



Рисунок 1.1 - Беспилотный автомобиль компании Google

В сегодняшний период установлено огромное число международных планов согласно исследованию беспилотных транспортных средств. Из их числа возможно отметить многочисленные проекты.

«Беспилотное транспортное средство (БТС) под названием Adaptive und Kooperative Technologien für den Intelligenten Verkehr (сокращенно АКТИВ) разрабатывают 28 известных европейских компаний. В их число вошли BMW, Siemens, Volkswagen, Bosch, Vodafone и еще 23 компании»[4]. Ценностью для основателей АКТИВ стали далее приведенные задачи: активная безопасность, автотранспортная коммуникация и организация регулировки автотранспортных потоков, предотвращение и борьба с авто заторами. Кроме того целью для АКТИВ считается концепция слежения за безопасностью в пути, позволяющая «обезопасить» водителя в сложной ситуации. В этом периоде АКТИВ станет осуществлять контроль

над плавностью разгона и торможения, синхронизируя данные действия, в том числе и для всей категории автомобилей. Помимо этого, она содержит аварийную систему торможения при внештатных ситуациях. Автомобиль АКТIV при тестированиях в беспилотном режиме на автодроме формирует скорость вплоть до 180 км/ч.

Иной немецкий проект по разработке беспилотного автомобиля под названием Leonie. Его концепция управления способна осуществлять контроль состояние 12 автомобилей в общем автотранспортном потоке вследствие большому числу датчиков, дальномеров и тепловизоров. Автомобиль-робот обладает возможностью перемещаться как в пустынной местности, так и в оживленных магистралях, одолевая крупные дистанции и тоннели. Кроме этого, германские конструкторы ведут исследования согласно формированию ещё 1-го беспилотного автомобиля-робота, который имеет наименование Made in Germany (MIG). План MIG исполняется только лишь в Германии, и в его проектировании принимают содействие научные работники с многих институтов. В настоящий период авто MIG способен осуществлять контроль над окружающей ситуацией в радиусе 70 м и обучается езде по перекресткам и пешеходным переходам. В качестве сенсорных систем применяются все эти же датчики и дальномеры.

В конференции HAVEit германское объединение Volkswagen показало технологические процессы Temporary AutoPilot «временный автопилот» на базе автомобиля Volkswagen Passat. Аналогично самолетам, водитель данного автомобиля обладает правом включать функцию автоматического управления машиной в незагруженных зонах пути. Временный автопилот Volkswagen функционирует при перемещении на магистралях с быстротой до 120 км/ч. Превосходство данной концепции управления состоит в том, что она уже практически расположена к внедрению на массовых автомобилях. Создание Volkswagen считается постепенным совершенствованием существующих технологий, с какими комплектуются современные серийные

машины, к которым принадлежат, в частности, адаптационный круиз контроль и систем слежения за полосой перемещения.

Автопилот машины Volkswagen способен руководствоваться согласно собственной полосе перемещения и сохранять устойчивую дистанцию за передвигающимся спереди автотранспортным средством и в случае потребности уменьшать скорость движения (рисунок 1.2).



Рисунок 1.2 - БАС, разрабатываемый фирмами Volkswagen и Bosch, Vodafone

Эта концепция, кроме того, может различать дорожные знаки и отвечать на их изменение скорости движения. Кроме магистралей автопилот Volkswagen способен перемещаться в заторах, повторяя однородный цикл старт – остановка. Однако, передвижение данного беспилотного автомобиля в сегодняшнее время ограничены, и он не способен преодолеть

установленный маршрут в отсутствии «поддержки» водителя в отличие от движений автомобилей фирмы Google.

Руководитель японской автомобилестроительной фирмы Nissan Карлос Гон выступил с заявлением о том, что приблизительно 2020 годы намечается массовое изготовление автомобилей с беспилотным управлением. Сведения о изготовлении беспилотных машин была объявлена управляющим Nissan в Калифорнии в городе Ирвайне, в котором разместился центр Nissan. Формироваться роботизированные автомобили станут с использованием технологии «Autonomous Drive» (рисунок 1.3), чтобы управлять таким автомобилем, нет необходимости присутствия водителя.

Новейшая создание станет содержать в себе лазерные радары, а кроме того камеры, сведения с которых станет попадать в компьютер, стремительно приспособляющийся к текущей дорожной ситуации.



Рисунок 1.3 - Беспилотный автомобиль компании NISSAN

С 2012 г., согласно заданию Минпромторга, ФГУП «НАМИ» осуществляет НИР согласно формированию умственных самоуправляемых беспилотных грузопассажирских АТС гражданского направления. В рамках плана разрабатывается образцовый пример «беспилотного» АТС в основе автомобиля «Лада-Калина» с механической коробкой передач (рисунок 1.4).

В структуру технологического зрения вступает комплект камер, радаров дальнего и ближнего воздействия, лазеры и GNSS приемники. С поддержкой концепции технологического зрения БАС согласно разрабатываемым методам и программному обеспечению исследует ситуацию вокруг себя на дистанции вплоть до 200 м, распознает дорожную разметку, дорожные знаки, светофоры, устанавливает постоянные и динамические объекты. Согласно записанному маршруту, а кроме того с помощью концепции GNSS, авто уже способен перемещаться вне дорог всеобщего использования с места А в место Б в отсутствии участия водителя.



Рисунок 1.4 - Макетный образец БАС на базе автомобиля «Лада-Калина»

«На макетном образце автомобиля «Лада-Калина» обеспечена работоспособность приводов педали тормоза, дроссельной заслонки, рулевого управления и селектора переключения режимов работы АККП, управляемых с помощью электронных пультов системы управления движением»[6].

С поддержкой концепции технологического зрения, поступающей в структуру концепции управления БАС, в то же время находят решение ключевые задачи увеличения безопасности дорожного движения. К примеру, в макетном стандарте БАС отрабатываются последующие концепции: предотвращения выезда на встречную полосу движения; определения дорожных знаков, выявления преград в пути (автомобиль, человек, животное и т.д.) и аварийной остановки машины.

Первостепенная концепция считается особенно важной для увеличения безопасности дорожного движения, т.к. приблизительно 80 % смертельных случаев при ДТП в Российской Федерации совершаются вследствие выезда АТС на встречную полосу движения. Метод и программное предоставление концепции предотвращения выезда на встречную полосу движения рассчитываются с учетом алгоритмов и программных обеспечений концепций определения дорожных знаков, выявления преград в пути.

«Введение «беспилотных» автомобилей позволит эффективно решать задачи повышения безопасности АТС, снижения числа пробок на дорогах, ДТП, травм и смертей, снижения расхода топлива, выброса вредных веществ, парниковых газов в атмосферу и повышение уровня комфорта для пассажиров»[8].

Базисные промышленные постановления при исследовании беспилотных машин имеют все шансы быть адаптированы и внедрены на серийных АТС. Беспилотный автомобиль считается многообещающим проектом с целью гражданского и военного направления.

1.2 Система автоматической и автономной парковки автомобиля

Автономная система только начинает поиски своих поклонников, в отличие от автоматической, которая их уже нашла. Ниже будут описаны разновидности системы и их составляющие.

Говоря о том, что автомобиль самостоятельно может припарковаться, на сегодняшний день, никого не удивишь. Чаще она известна как автоматическая парковка автомобиля. Прогресс не стоит на месте и компания BMW шагнула вперед, опережая всех, предложив автономную систему парковки Remote Valet Parking Assistant. По-другому это дистанционный помощник парковки.

Но в основе таких прогрессирующих систем лежат более старые, рабочие образцы. В данном случае это автоматическая система парковки, которая часто встречается на многих современных автомобилях, даже и не новых.

Система автоматической парковки (другое наименование - интеллектуальная система помощи при парковке), является активной парковочной системой, так как обеспечивает парковку автомобиля в автоматическом или автоматизированном (выполняются отдельные функции) режиме.

Эта система может использоваться при параллельной или перпендикулярной парковке. Наиболее часто распространяются системы с параллельной парковкой. За основу парковки берется скорость движения автомобиля и угол поворота колеса.

В разных компаниях система называется по-разному:

- Volkswagen - Park Assist, PAV;
- Lexus или Toyota - IPAS;
- Opel - Advanced Park Assist;
- Ford или Mercedes-Benz - APA;
- BMW - RPA System.

Главными составляющими являются ультразвуковые датчики, далее идут выключатели, блок управления электронный и другие исполнительные устройства машины.

Важно, что без дополнительных систем автоматическая парковка не сможет работать.

Ультразвуковые датчики в системе аналогичны к пассивной системе парковки, но отличаются большей дальностью действия, это порядка до 4.5 м. Зависимо от производителя системы и марки автомобиля, количество датчиков может отличаться. Как правило, устанавливается 12 датчиков ультразвуковых, четыре сбоку автомобиля, четыре спереди и четыре позади автомобиля.

В действие система приводится принудительно, включением кнопки на панели приборов, или рулю управления, когда водитель планирует припарковаться. Поле активации системы, электронный блок начинает принимать сигнал от ультразвуковых датчиков, преобразуя их в управляющие функции для исполнительных устройств. В качестве таких устройств можно назвать управление двигателем, электроусилитель руля, автоматическая или механическая кпп и система курсовой устойчивости.

В зависимости от производителя, информация может выводиться на дисплей, как вспомогательная для водителя и параллельно в автоматическом режиме управлять автомобилем, если водитель дал на то разрешение.

Принцип поиска места для парковки. Как говорилось ранее, поиск подходящего места для парковки осуществляется с помощью датчиков ультразвука. По бокам расположены четыре таких датчика по две на каждую сторону.

Скорость движения автомобиля, между рядами припаркованных автомобилей, достигает 20 км/час для поперечной парковки, 40 км/час для параллельной парковки. Датчики сканируют расстояние между машинами и их положение относительно вашего автомобиля.

Автоматическая парковка автомобиля. Данный процесс парковки можно поделить на два вида:

- с помощью инструкций под управлением водителя;
- автоматическая парковка без участия водителя.

В первом способе для водителя будут выводиться подсказки, визуальные и тестовые инструкции на информационный дисплей. На основе подсказок и траектории водитель принимает решение, как лучше припарковаться. Как правило, это рекомендации по рулевому колесу, какой угол поворота лучше выбрать и направление по движению.

Второй способ отличается, в сравнении с предыдущим водитель не участвует в процессе парковки, а все функции выполняются в автоматическом режиме. Вместо водителя руль будет поворачиваться самостоятельно в нужном направлении, так же как и нажатие педали газа.

В целях безопасности автоматическую систему парковки в любое время можно перевести в режим ручной парковки (рисунок 1.5).



Рисунок 1.5 – Режим ручной парковки

Развитие не стоит на месте, и вспомогательные системы автомобиля так же прогрессируют. Как уже говорилось, автоматическая система может работать, когда водитель сидит и наблюдает за процессом работы парковки автомобиля.

Система автономной парковки автомобиля состоит из электронного блока управления, устройств исполнения команд и входных устройств. К приборам для считывания или входным относят лазерные датчики (лидары) и пульт дистанционного управления.

В основе лежит система автоматической парковки автомобиля и предупреждение столкновения по всему периметру машины. Благодаря таким составляющим и технологии значительно экономится время, лучше всего система показала себя в многоэтажных паркингах. Кроме того система может поместить автомобиль в пространство ограниченное 20 см. с каждой стороны машины, что очень экономит пространство.

Принцип работы автономной системы. Активация системы автономной парковки автомобиля начинается с нажатия кнопки на пульте дистанционного управления(рисунок 1.6).



Рисунок 1.6 – Пульт дистанционного управления Smart Watch

Самым главным отличием такой системы от предыдущего поколения является самостоятельная парковка без участия водителя. Так же автомобиль может самостоятельно выдвигаться к месту посадки водителя и пассажиров.

Принцип работы и последовательность действий системы следующие. Лидары, сняв данные, пересылают их на электронный блок управления. Программа, зашитая в блок, пропускает полученную информацию через алгоритм для оценки ситуации, препятствий, регулировки скорости, навигации и маршрутка поиска места.

Благодаря использованию множества систем автомобиля, блок управления сам может решить насколько разогнать или приостановить машину. Одним недостатков можно считать то, что в блок прошивается цифровая карта (план) здания паркинга, при этом система не использует GPS карты и сигнал. Чаще всего это делается для закрытых паркингов, где сигнал просто-напросто не может пройти. Но для открытых парковок прошивать нечего не надо.

В результате электронный блок управления формирует сигналы для управляющих систем, которые потом передаются на эти системы автомобиля: управление двигателем, коробкой передач, рулевым колесом и прочие. Огромным плюсом системы является то, что она не стоит больших денег, а ориентация в паркинге не займет много времени. Кроме того водитель не обязан сидеть и следить за автомобилем до последнего, пока автомобиль не остановится. Как результат, многие производители будут устанавливать автономную систему парковки на новые серийные автомобили вместо автоматической парковки.

1.3 Проблемы автопилота

Автомобиль Tesla, движущийся в автоматическом режиме, вновь совершил дорожно-транспортное происшествие (ДТП). Кроссовер Model X

врезался в ограждение справа от дороги, после чего отлетел и ударился о бетонный отбойник, разделяющий полосы встречного направления. Машина перевернулась, но жертв, к счастью, удалось избежать. По словам водителя, в момент ДТП автомобиль находился в режиме автопилота. За несколько недель до этого седан Model S под управлением автопилота совершил столкновение с прицепом фуры, что привело к смерти водителя.

Информация об авариях с автопилотируемыми Tesla приходит все чаще, а компания не устает объяснять, что электроника в машинах является вспомогательной, а ответственность за возможные происшествия в любом случае лежит на водителе. Тем не менее, информационный фон вокруг так называемого автопилота, равно как и заявления Илона Маска, создают у многих обманчивое впечатление того, что на электронику Tesla уже можно полностью положиться. Однако возможности автомобиля на данном этапе не позволяют использовать ее в беспилотном режиме.

Выделяют три практических сложности, с которыми сталкиваются автопилоты сейчас и которые не выглядят решаемыми в ближайшие годы.

Первая проблема, датчики. Основное затруднение формирования автопилотов не методы, а отвратительное представление автомобилем находящегося вокруг места. Данных с современных измерителей категорично мало и усовершенствование приспособления принятия решений (нейросети и так далее) немного что изменяет. Сканирующие лидары (темные ведерки, те что встают в основной массе прототипом, в том числе Google Car) за пределами дороги с целью массовых автомобилей и никак не предоставляют нужной полноты данных, с целью решительного перемещения.

Простая иллюстрация: AI побеждает пилота на симуляторе воздушного боя в той ситуации, когда у машины есть полная информация о происходящем. Компьютер проедет вас в любом симуляторе. На дороге этого не произойдет.

И компьютерные полигоны для автономного транспорта, которые обучаются на синтетической информации, ничего серьезно не изменят.

Есть мнение, что при существующих датчиках (камеры, радары, лидары и т.п.) полностью решить проблему самоуправления вообще нельзя. Так что совершенствовать возможности современных датчиков, придумывать новые для корректной «оцифровки» пространства в динамике.

«На сегодняшний день одна из проблем навигационных устройств – это то, что они не ведут пользователя по полосам. Эта проблема увеличивает время в пути, пробки и аварийность. Недавно google maps начали отображать разметку дороги перед поворотом, что уже хороший результат, но и тут можно многое улучшить. Карты не знают, на какой полосе сейчас находится машина, средствами gps узнать это проблематично, у gps слишком большая погрешность для этого»[14]. Если бы мы знали текущую полосу, то знали бы скорость движения по полосам и могли бы задолго подсказывать пользователю в явном виде, на какую полосу и когда ему лучше перестроиться(рисунок 1.7).

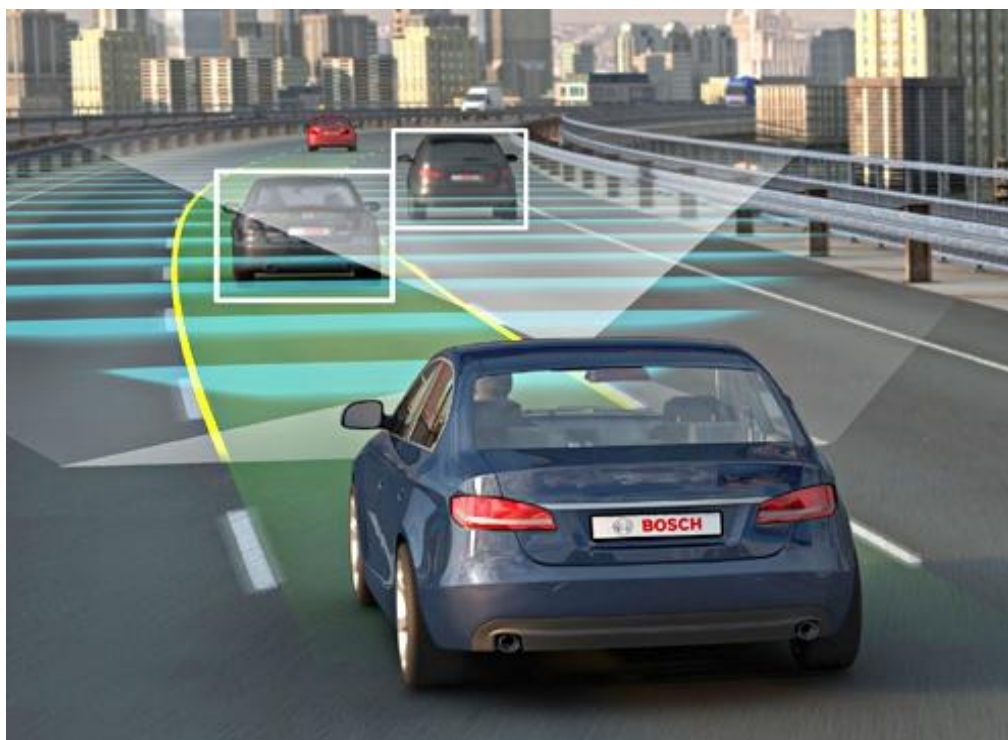


Рисунок 1.7 – Датчики автопилотов

Чем-то похоже на вопрос с аккумуляторами. Пока емкость аккумуляторов не получится повысить в разы, а лучше на порядок, носимая электроника, электрокары и многие другие топчутся на месте.

Вторая проблема, не универсальность. Все знакомы с Законом Парето, так вот в автопилотах он тоже выполняется. 80% дорожных ситуаций машины понимают, оставшиеся 20% все портят.

«Это не значит, что автопилоты могут ехать по 80% дорог в мире. Это означает, что в рамках любой произвольно взятой поездки 80% ситуаций не вызовут проблем, 20% — вызовут. Опасная пропорция»[14].

Эти задачи и окажутся самыми простыми. Автобанный автопилот, автопарковка, заезд в гараж — не без помарок, но функционирует.

Ученым осталось самое сложное: проезд пересечений, перестроения в оживленном потоке и еще несколько десятков похожих сценариев. С некоторыми кейсами не очень понятно, что вообще делать. Например, автомобилю сложно заранее понять, проедет ли он между двумя близко расположенными препятствиями. И пока невозможно понять свой динамический коридор (то есть границы пространства, которые автомобиль занимает в динамике), а без этого невозможно корректно просчитать дорожную ситуацию.

Существующие автопилоты настолько не универсальны, что требуют каждый дорожный сценарий описывать отдельно. Читали комментарий Mobileye к пресловутой аварии? Официальная позиция такова: система предотвращения столкновений не понимает приезда автомобиля со встречной полосы, это будет обеспечено к 2018 году.

«Для решения данной проблемы клиенту, как минимум, не стоит верить эффектным демонстрациям на автострадах. Что до игроков, то они пока не понимают, какой путь верен. Отсюда нейросети, ручная разметка сценариев (пишут, что в Mobileye этим занимаются 600 человек, а скоро будет более 1000), виртуальные полигоны и тому подобное» [14].

Проблема третья, рыночная. Автопилоту мало быть функциональной, нужно стать дешевой. Автомобильной навигации больше двадцати лет. Но при цене системы в пару-тройку тысяч долларов она до сих пор ставится менее чем на 20% всех новых автомобилей. Самоуправляемый функционал еще дороже.

«За лидар Velodyne 70 тысяч долларов или 30 тысяч за несколько лидаров Sick это очень много. Если ориентироваться на серийные автомобили, advanced driver assistance systems, сокращенно ADAS, (которые еще не автопилоты) добавляют к цене 7-12 тысяч долларов» [17].

Исследование Boston Consulting Group говорит, что потратить на автопилот более 5 тысяч долларов готовы только 17% покупателей (американских покупателей). То есть, чтобы завоевать хотя бы 15 процентов рынка, системе автопилотирования нужно стать, грубо говоря, на порядок умнее и вдвое дешевле, чем сейчас. Это не выглядит близкой перспективой.

Так что помощников с элементами автопилотирования (автопарковка, автодвижение в пробке, движение по трассе с разметкой или даже конвойные режимы) в ближайшие пару-тройку лет будет все больше. Автопроизводители удешевят и масштабируют эти функции. Toyota предлагает уже свой простейший Safety Sense всего за 500-1000 долларов.

А в сложных ситуациях (то есть самые 20 процентов поездки) управление на долгие годы останется на человеке.

«Привыкать к аббревиатуре ADAS, которая в ближайшие пять лет станет такой же привычной как стали ABS и ESP. После 2020 года стоит ждать активного госрегулирования в этой области, системы активной помощи станут обязательными в Европе и США»[14].

И понимать, что полноценного автопилота пока на рынке нет, и вообще у них на данный момент неустранимые проблемы.

Выводы

Произведен анализ современного состояния беспилотного автомобиля. Рассмотрены существующие автономные транспортные средства и способ их работы. Были изучены системы автоматической и автономной парковки автомобиля. Произведен анализ проблем возникающих у автопилота.

Выявлены следующие проблемы:

1 Датчики. Главное замедление развития автопилотов не алгоритмы, а отвратительное понимание машиной окружающего пространства. Информации с современных датчиков категорически недостаточно и совершенствование механизма принятия решений (нейросети и так далее) мало что меняет. Сканирующие лидары (черные ведерки, что стоят на большинстве прототипов, включая Google Car) за пределами дороги для серийных автомобилей и не дают необходимой для уверенного движения полноты информации.

2 Не универсальность. Аналогично Закону Парето, в автопилотах он тоже выполняется. 80% дорожных ситуаций машины понимают, оставшиеся 20% все портят.

3 Рыночная. Для автопилота важно быть не только функциональным, но и дешевым. Автомобильную навигацию создали больше двадцати лет назад. Но при цене системы в пару-тройку тысяч долларов она до сих пор ставится менее чем на 20% всех новых автомобилей. Самоуправляемый функционал еще дороже.

Рассмотрены условия выхода на рынок автономного транспорта и методы снижения их стоимости.

2 Разработка алгоритмов управления автономными транспортными средствами

2.1 Система навигации

Главными задачами системы навигации являются:

- определение нахождения автомобиля в текущий момент
- ввод пункта назначения с определением оптимального маршрута

При выборе маршрута предлагается 3 варианта – быстрый, нормальный и короткий. Кроме этого указывается, где находятся платные дороги и особенности движения по каждому маршруту.

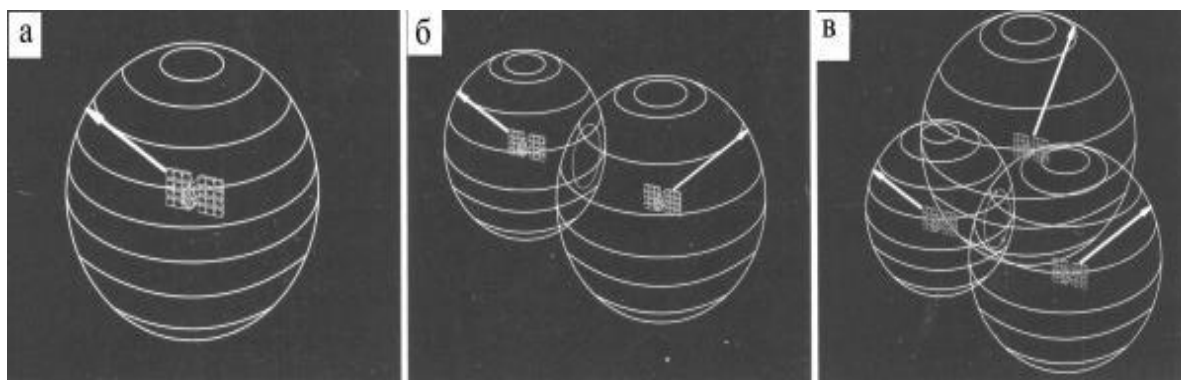
Для того чтобы пользоваться GPS-навигацией, нужен как минимум GPS-приемник. Но сам по себе он не более чем спутниковый компас, знающий свои точные координаты. Чтобы связать эти координаты с конкретной цифровой картой местности, нужно более сложное устройство, например, GPS-навигатор со встроенным GPS-приемником

«GPS-навигаторы– устройства со сравнительно простой программной оболочкой, ориентированные преимущественно на решение навигационных задач и способные работать только с одним предусмотренным производителем типом карт» [16].

Вычисление положения GPS-приемника осуществляется на основе заранее известных координат спутников системы. Физически это выражается в том, что исходными данными для решения задачи позиционирования являются расстояния от объекта до всех видимых им в данный момент спутников. Для упрощения допустим, что все видимые спутники находятся на своих орбитах в неподвижном состоянии.

Геометрия вычисления положения. «Искомый объект находится на поверхности сферы, описанной вокруг точки местоположения сателлита и имеющей радиус, равный расстоянию до объекта. Для определения местоположения контакта с одним спутником недостаточно. Пересечение сфер от двух спутников дает окружность – задача из трехмерной

превратилась в двухмерную» [16]. Теперь известно, что приемник находится где-то на полученной окружности. Сфера, очерченная вокруг третьего видимого спутника, пересекает окружность в двух точках, одна из которых является паразитной, так как находится либо в недрах земного шара, либо в верхних слоях атмосферы(рисунок 2.1).



а – сфера от одного спутника; б – пересечение сфер от двух спутников;

в – пересечение сфер от трех спутников

Рисунок 2.1 - Обнаружение объекта спутниками

«Оставшаяся точка и характеризует координаты приемника. Расстояния до спутников (радиусы описанных сфер) вычисляются просто – на основе фиксации времени прохождения сигнала до объекта и его скорости» [13].

Для определения положение спутников на орбите помимо совокупности спутников, рассредоточенных по стационарным орбитам, существует наземный комплекс управления. В его состав входят станции слежения, поддерживающие постоянный контакт с элементами орбитальной группировки. По полученным данным в центре управления вычисляются точные координаты искусственных спутников и через станции связи передаются на летательные аппараты. «При расчетах принято, что скорость распространения сигнала равна скорости света. Для этого необходимо учесть еще точность и синхронизацию работы часовых механизмов, которыми оснащены спутник и приемник, а также искажения, вызванные различными

препятствиями на пути прохождения информационной волны» [17]. Для устранения ошибок в компьютере приемника используются специальные алгоритмы, которые корректируют время до тех пор, пока местоположение приемника не будет определено с заранее заданной погрешностью. Алгоритм также учитывает данные, поступившие от четвертого, пятого и других спутников, которые находятся в «зоне видимости» приемника.

Отметим, что полноценная группировка, которая обеспечит покрытие всей поверхности земного шара, должна включать 24 орбитальных объекта, то есть максимальное количество видимых приемником спутников в любой точке земли – 12 единиц. Однако на сегодняшний день количество действующих аппаратов систем навигации уже составляет 30 единиц.

На рисунке показана структура навигационной системы. Система может осуществлять навигационное счисление, определять положение автомобиля на карте местности по конфигурации пройденного пути, определять абсолютные координаты с помощью спутниковой системы GPS. С помощью навигационного счисления определяют относительное положение автомобиля и направление движения по информации, полученной с датчиков скорости вращения колес и азимута.

Конфигурация участка пройденного пути, полученная с помощью навигационного счисления, сравнивается с конфигурацией дорог, нанесенных на карту. Определив дорогу, по которой движется автомобиль, система находит и его текущие координаты. Более точное определение координат автомобиля на карте производится с помощью GPS по широте и долготе. Считается, что для практических целей достаточно знать координаты автомобиля с точностью до размера половины квартала, т. е. ± 100 метров.

Автомобильная навигационная система должна иметь в своем составе датчики пройденного пути и направления движения.

Датчик пройденного пути — это та или иная конструкция электронного одометра, информация в который поступает с датчиков скорости вращения

колес ABS. Одометрам присущ ряд систематических погрешностей, которые необходимо корректировать. К ним относятся:

1 Разница в диаметрах новой и изношенной покрышки дает погрешность в определении пройденной дистанции до 3%.

2 За счет увеличения диаметра покрышки от центробежной силы на каждые 40 км/час скорости автомобиля погрешность в определении пройденной дистанции увеличивается на 0,1...0,7%.

3 Изменение давления в шинах на 689 кПа увеличивает погрешность на 0,25...1,1%.

Для определения направления движения автомобиля обычно используются датчик азимута, датчики скорости вращения колес, гироскопы (рисунок 2.2).



Рисунок 2.2- Структура навигационной системы

2.2 Обнаружение столкновений

«Обнаружение столкновений» (англ. Collision detection) — вычислительная проблема обнаружения пересечений между собой двух или

больше объектов. Тема чаще всего связана с её использованием в физических движках, компьютерной анимации и робототехнике. В дополнение к определению, столкнулись ли два объекта, системы обнаружения столкновений могут вычислить время воздействия и сообщить о коллекторе контакта (набор пересечения точек)» [14]. Ответ на столкновение (что происходит, когда столкновение обнаружено) зависит от используемого моделирования(рисунок 2.3).

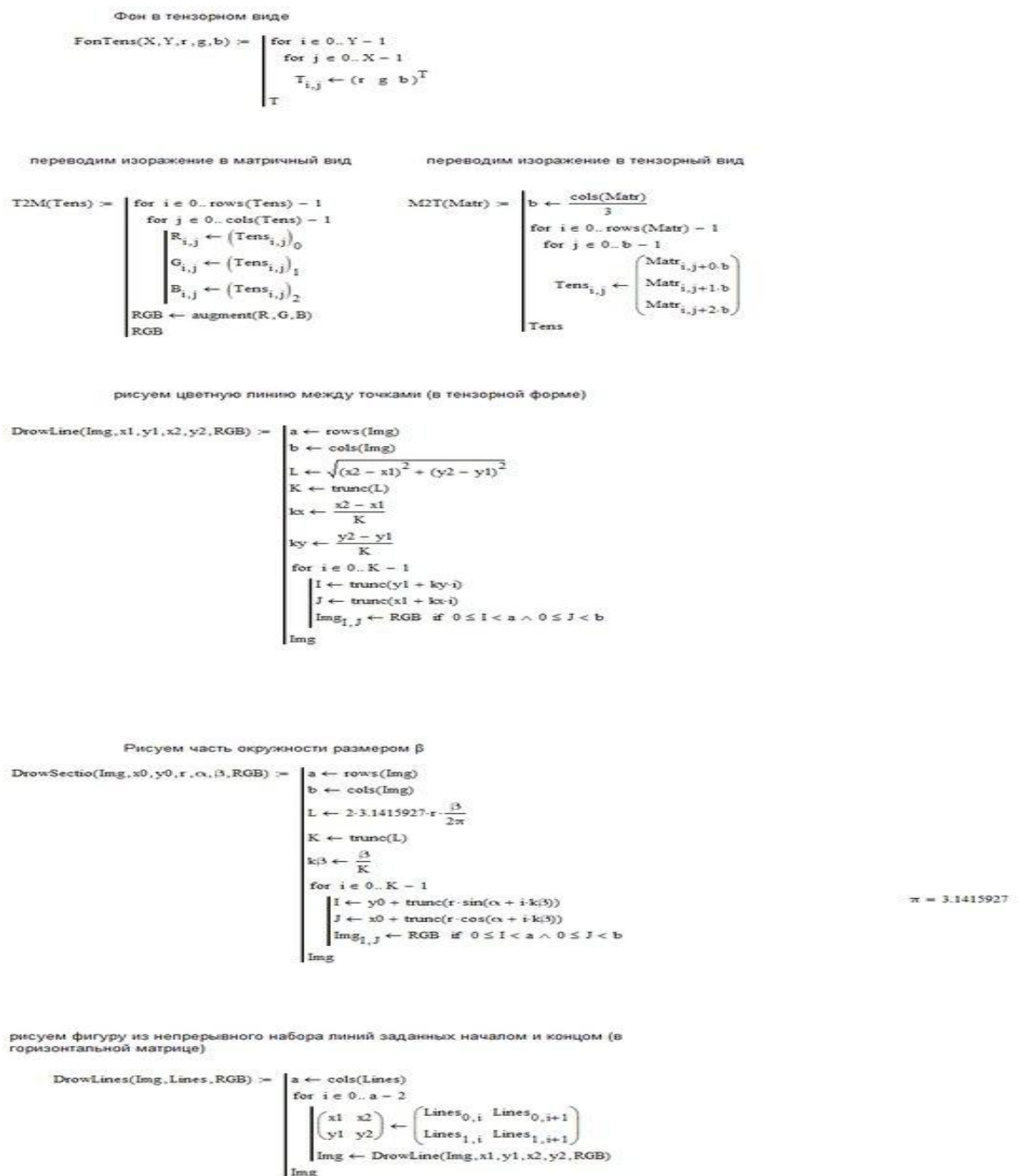


Рисунок 2.3 – Поле коллизии

Решение проблем обнаружения столкновений требует широкого применения понятий из линейной алгебры и вычислительной геометрии. Алгоритмы обнаружения столкновений являются одним из основных составляющих трёхмерных компьютерных игр. Коллизия определяет физическую вещественность объектов в моделировании. Например, автомобиль, помещенный на автомобильную дорогу без коллизий, просто провалится сквозь движущийся транспорт. Или открытая дверь автомобиля без коллизии также пройдет через любой объект, а не среагирует на столкновение правильно

Вводится поле коллизий, массив где каждый компонент представлен в виде бинарной графики (своих естественных заданных размеров и форм), далее будет использована для более точного обнаружения столкновений пилота с другими пилотами и\или стенами (рисунок 2.4).

Далее описывается работа с векторами: отрисовка векторов, сенсоры, отрисовка сенсоров, обнаружение коллизий, формирование поля коллизий, модуль перемещений и принятия решений, главный цикл игры.

Рисуем луч цвета RGB из точки **PointSens** в направлении вектора **VectSens**, до конца поля отрисовки **Fon** (в тензорном виде)

```
DrowSensBeam0(PointSens, VectSens, Fon, RGB) :=
  a ← rows(Fon)
  b ← cols(Fon)
  A ← AngleVect(VectSens)
  (x1 y1) ← (PointSens0 PointSens1)
  dia ←  $\sqrt{a^2 + b^2}$ 
  (x2 y2) ← (x1 + dia·cos(A) y1 + dia·sin(A))
  Fon ← DrowLine(Fon, x1, y1, x2, y2, RGB)
  Fon
```

Рисуем луч цвета RGB из точки **PointSens** в направлении вектора **VectSens**, до точки соприкосновения луча (на расстояние **ValueSensBeam**) с объектом или границей (в тензорном виде)

```
DrowSensBeam1(PointSens, VectSens, ValueSensBeam, Fon, RGB) :=
  A ← AngleVect(VectSens)
  (x1 y1) ← (PointSens0 PointSens1)
  VecA ← (ValueSensBeam A)T
  VecK ← VA2VK(VecA)
  (x2 y2) ← (x1 + VecK0 y1 + VecK1)
  Fon ← DrowLine(Fon, x1, y1, x2, y2, RGB)
  Fon
```

Рисунок 2.4 – Отрисовка лучей сенсоров

Отрисовка нужна для контроля за правильностью работы алгоритмов. Используя библиотеку отрисовки графики, описывается отрисовка лучей сенсоров

Данный рисунок отображает: 0-вариант рисует просто луч из точки, 1-вариант рисует отрезок в направлении луча из точки начала луча до соприкосновения с объектом или границей.

ValueSensBeam – значение, которое выдала функция SensBeam, определяющая расстояние до объекта или границы.

SensBeam - сенсорный луч.

Выдает "true" при наличии соприкосновении с объектом и "false" если столкновений не обнаружено (рисунок 2.5).

```

KollizCircle(PointObj, FeeldKolliz, RCircle, RQuadrat) :=
  Rq ← ceil( $\frac{RQuadrat}{2}$ )
  a ← rows(FeeldKolliz)
  b ← cols(FeeldKolliz)
  Kollizia ← "false"
  for i ∈ -Rq..Rq
    for j ∈ -Rq..Rq
      I ← PointObj1 + i
      J ← PointObj0 + j
      if 0 ≤ I < a ∧ 0 ≤ J < b
        LObj ← if (FeeldKolliz1,J ≠ 0,  $\sqrt{i^2 + j^2}$ , Rq)
        Kollizia ← "true" if LObj ≤ RCircle
        break if Kollizia = "true"
  Kollizia

```

Рисунок 2.5 – Определение столкновений

Подразумевается вот такой круг с таким квадратом близости, то есть обрабатываются точки внутри квадрата (чтоб производительность была выше) и досрочное прерывание и выход из цикла, если коллизия обнаружена (рисунок 2.6).

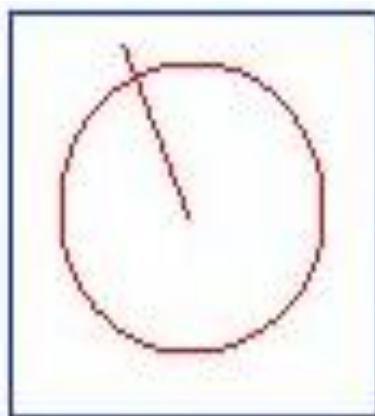


Рисунок 2.6 – Обнаружение коллизии

Модуль формирования поля коллизий FeeldKolliz (рисунок 2.7)

Формируем поле коллизий **FeeldKolliz** используя список объектов-пилотов **ListObj** (на случай если пилотов несколько) и файл поля\трассы **RacingWay**

```

FeeldKolliz(ListObj, RacingWay) :=
  c ← rows(ListObj)
  Feeld ← RacingWay
  for i ∈ 0..c - 1
    x0 ← (ListObji,0)0
    y0 ← (ListObji,0)1
    RCircle ← ListObji,1
    Feeld ← DrawCircleKolliz(Feeld, x0, y0, RCircle)
  Feeld

```

Рисунок 2.7 – Модуль формирования поля коллизий

Так выглядит список ListObj (рисунок 2.8)

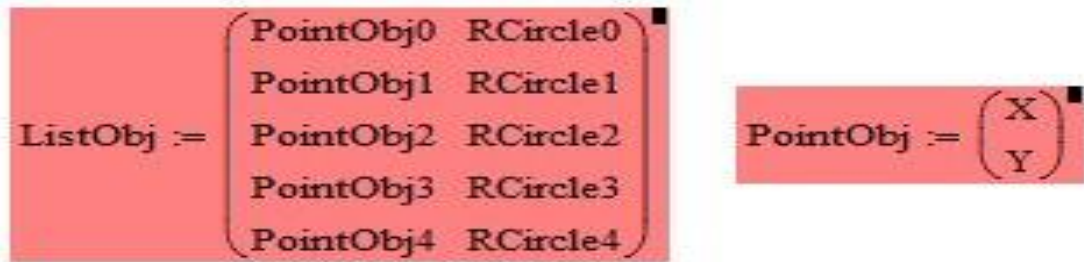


Рисунок 2.8 - список ListObj

Пример, рисуем три круга разного радиуса на трассе (белым обозначены препятствия, объекты; черным - свободные области) (рисунок 2.9)

```
ListObj0,0 :=  $\begin{pmatrix} 20 \\ 20 \end{pmatrix}$     ListObj0,1 := 10
ListObj1,0 :=  $\begin{pmatrix} 40 \\ 60 \end{pmatrix}$     ListObj1,1 := 30
ListObj2,0 :=  $\begin{pmatrix} 80 \\ 50 \end{pmatrix}$     ListObj2,1 := 20
D := FeeldKolliz(ListObj, Feeld)
```



Рисунок 2.9 – Пример образования поля коллизии

Описание модуля удаления объектов с поля коллизий, для правильной работы модуля поиска коллизий (рисунок 2.10).

Удаляем соответствующий объект (не трассу) с поля коллизий

```
DeleteObj(FeeldKolliz, ListObj, N) :=  $\begin{cases} x0 \leftarrow (ListObj_{N,0})_0 \\ y0 \leftarrow (ListObj_{N,0})_1 \\ RCircle \leftarrow ListObj_{N,1} \\ Feeld \leftarrow DrowCircleKolliz(FeeldKolliz, x0, y0, RCircle, 0) \\ Feeld \end{cases}$ 
```

Рисунок 2.10 - Описание модуля удаления объектов с поля коллизий

Пример удаления окружности с наибольшим диаметром (рисунок 2.11)

```
d := DeleteObj(D,ListObj,1)
```

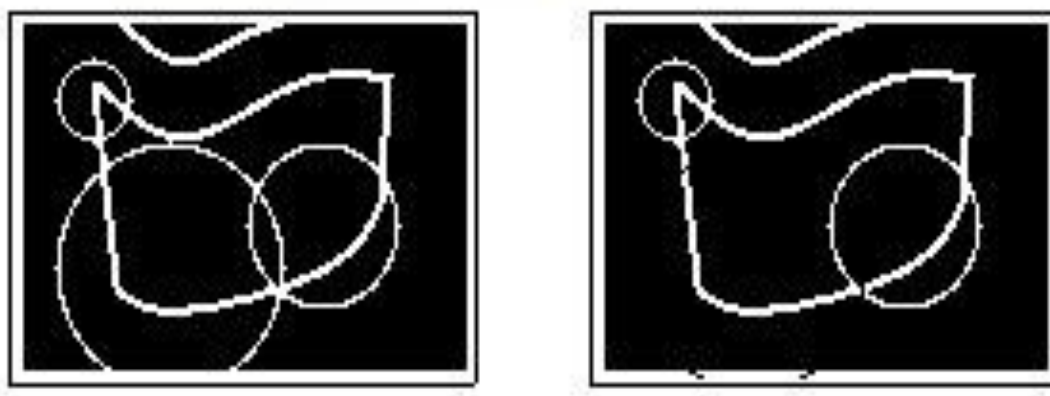


Рисунок 2.11- Пример удаления окружности с наибольшим диаметром

Предварительная модель перемещения и работы с сенсорами изображена ниже (рисунок 2.12).

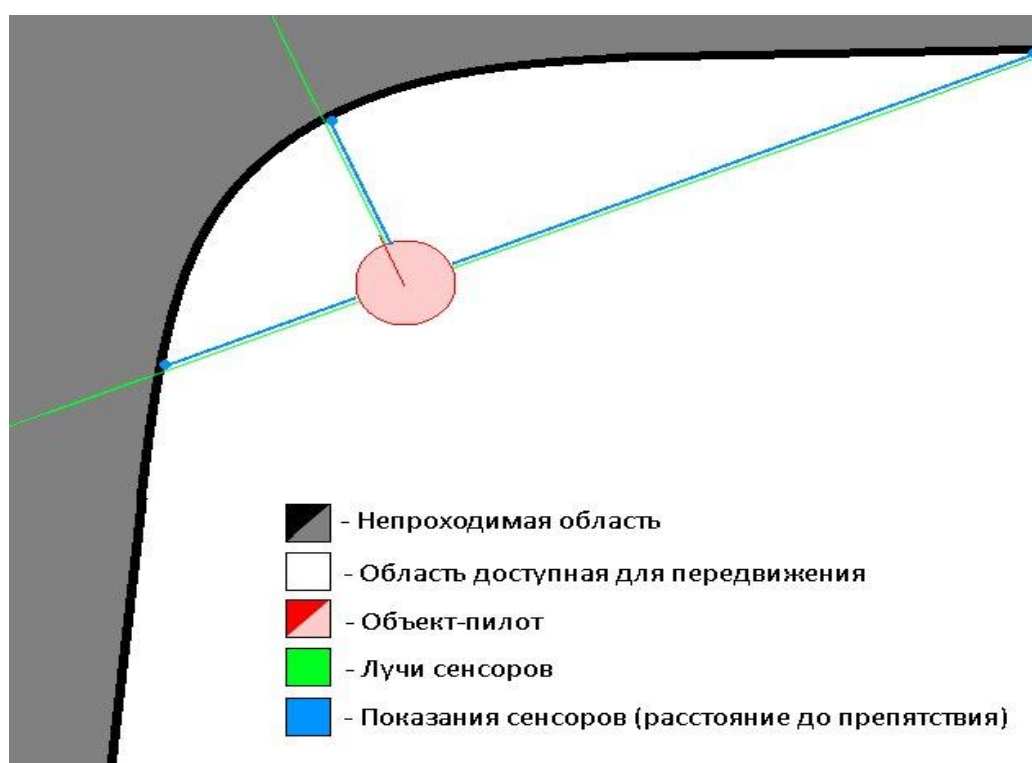


Рисунок 2.12 – Предварительная модель перемещения и работы с сенсорами

Пример изменения скорости в зависимости от работы нулевого сенсора (рисунок 2.13)

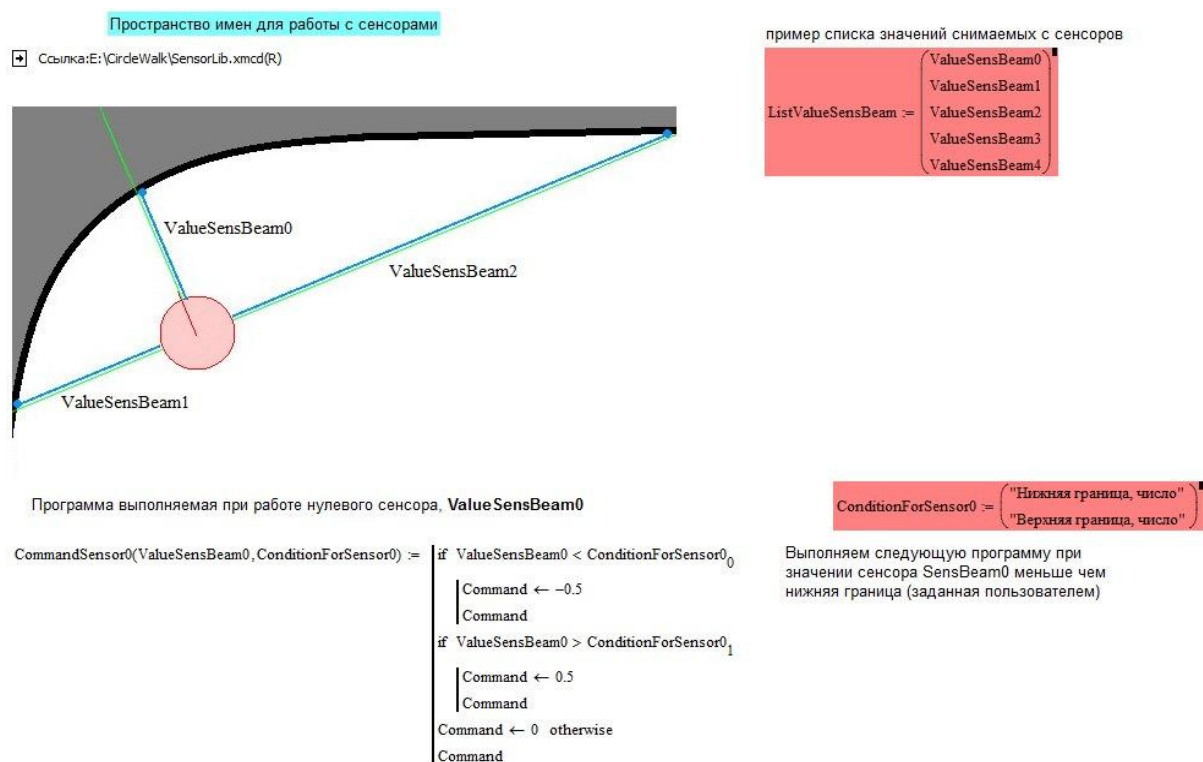


Рисунок 2.13 - Пример изменения скорости

Так меняется скорость в зависимости от работы нулевого сенсора расстояние до препятствия (рисунок 2.14).

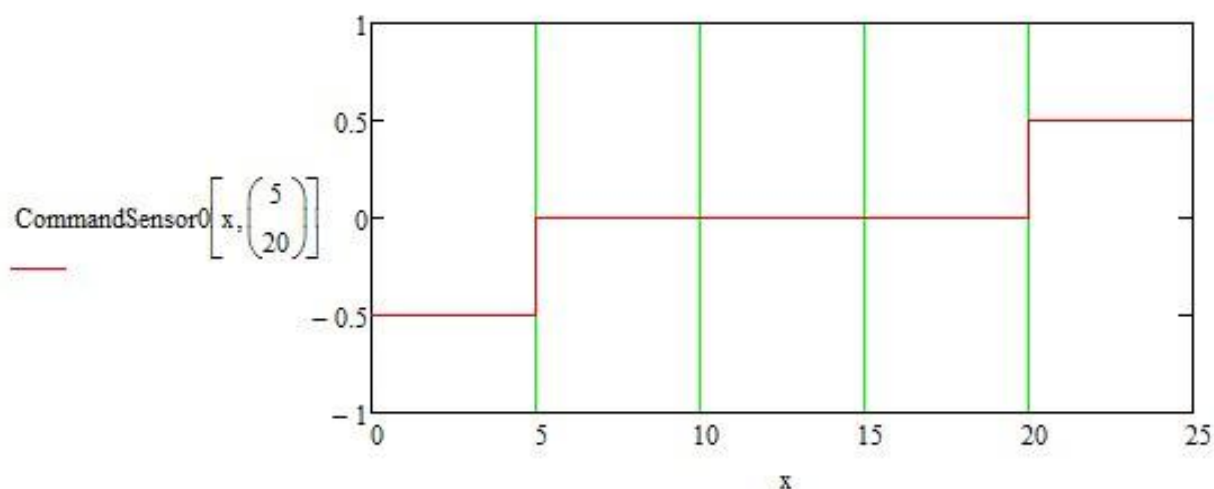


Рисунок 2.14 – График изменения скорости

- если расстояние меньше 5 метров, скорость убавляется на 0.5;
- если расстояние от 5 до 20 метров, то скорость не меняется;
- если расстояние больше 20 метров, скорость возрастает на 0.5.

Следует отметить, что скорость увеличивается или уменьшается постоянно, если выполняются условия. Поэтому получаем: равнозамедленное движение в первом случае, равномерное во втором и равноускоренное в третьем.

Далее представлен листинг и пример работы программы, зависящей от работы первого сенсора (рисунок 2.15), реализован поворот в зависимости от близости до препятствия (сбоку).

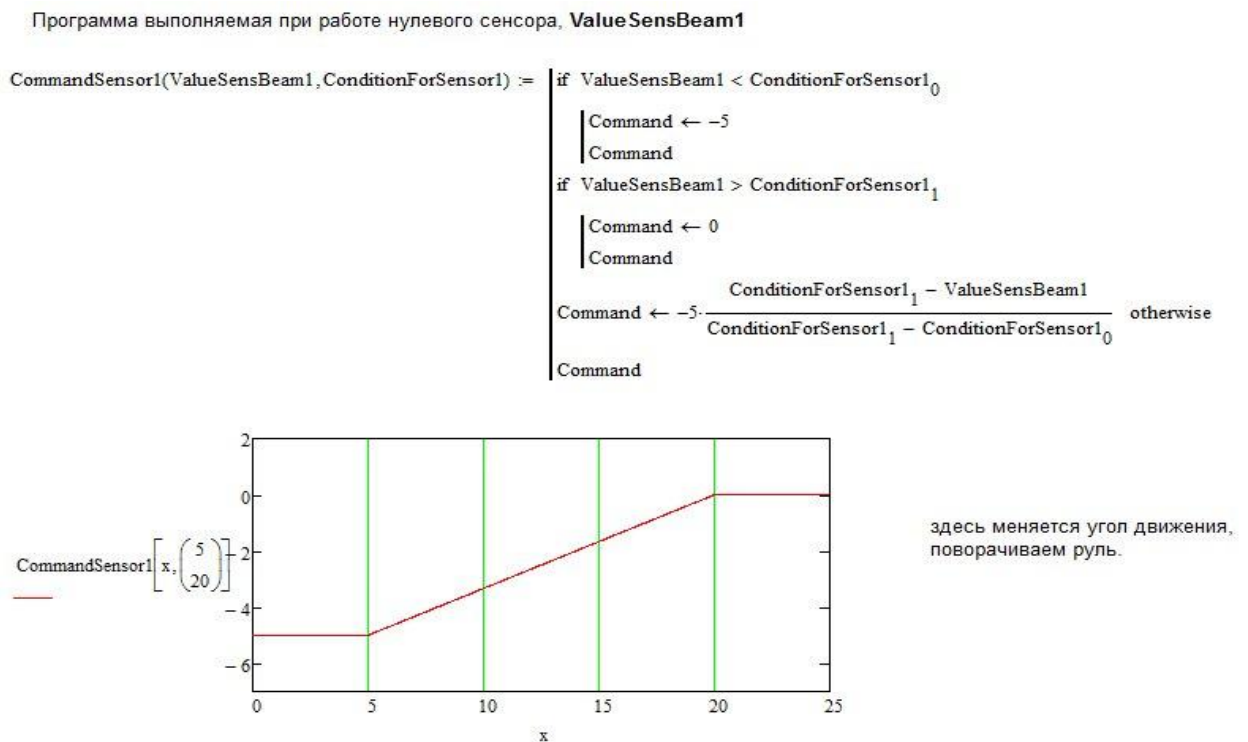


Рисунок 2.15 – Пример работы программы

Максимальный угол поворота принят за 5 градусов, и плавно и равномерно снижается до 0 при отдалении от препятствия (стены). Если препятствие дальше, чем в 20 метрах, пилот не подруливает.

Знак указывает на сторону поворота (на рисунке выше поворот реализован против движения часовой стрелки, поэтому имеет знак минус).

Тоже для второго сенсора поворот налево (рисунок 2.16).

Программа выполняемая при работе нулевого сенсора, **ValueSensBeam2**

```

CommandSensor2(ValueSensBeam2, ConditionForSensor2) :=
  if ValueSensBeam2 < ConditionForSensor20
  | Command ← 5
  | Command
  if ValueSensBeam2 > ConditionForSensor21
  | Command ← 0
  | Command
  Command ← 5 ·  $\frac{\text{ConditionForSensor2}_1 - \text{ValueSensBeam2}}{\text{ConditionForSensor2}_1 - \text{ConditionForSensor2}_0}$  otherwise
  Command
  
```

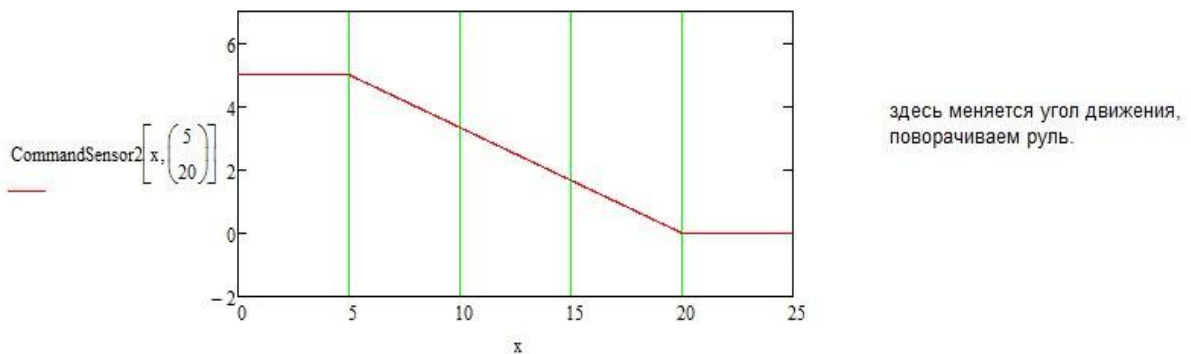


Рисунок 2.16 - Поворот налево для второго сенсора

Функция, вычисляющая следующее положение на основе текущего положения, текущего направления и скорости и вычисленного смещения на основе работы сенсоров. С ограничением по скорости и углу поворота и заданной дискретностью времени.

Тест работы сенсоров, направляющих векторов и отрисовки графики.

Зеленая линия - луч сенсора.

Синяя линия - расстояние до цели определенное этим сенсором.

(R) — означает, что путь к библиотеке задан относительно, следовательно, не зависит от местоположения папки CircleWalk на диске.

Пространство имен для использования функций всех написанных ранее библиотек (рисунок 2.17).

- Ссылка: E:\CirdeWalk\VectorLib.xmcd(R)
Пространство имен для работы с векторами
- Ссылка: E:\CirdeWalk\DrawLib.xmcd(R)
Пространство имен для работы с элементами графики
- Ссылка: E:\CirdeWalk\SensorLib.xmcd(R)
Пространство имен для работы с сенсорами
- Ссылка: E:\CirdeWalk\KollizLib.xmcd(R)
Пространство имен для работы с элементами обнаружения коллизий
- Ссылка: E:\CirdeWalk\CalculeLib.xmcd(R)
Пространство имен для работы с функциями "принятия решений"
- Ссылка: E:\CirdeWalk\PhysicLib.xmcd(R)
Пространство имен для работы с физикой и перемещениями

Рисунок 2.17 - Пространство имен для использования функций всех написанных ранее библиотек

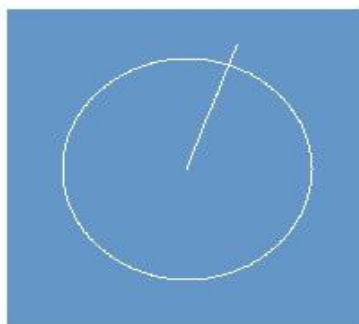
Отрисовка объектов для визуализации (рисунок 2.18).

Рисуем объект (окружность со стрелкой\линией). VectObj задан в угловых координатах

```

DrawObj(PointObj, VectObj, RCircle, Img, RGB) :=
  Img ← DrawSectio(Img, PointObj0, PointObj1, RCircle, 0, 2π, RGB)
  VectObj2 ←  $\begin{pmatrix} RCircle \cdot 1.2 \\ VectObj_1 \end{pmatrix}$ 
  Vect ← VA2VK(VectObj2)
  PointCur ← PointObj + Vect
  Img ← DrawLine(Img, PointObj0, PointObj1, PointCur0, PointCur1, RGB)
  Img

```



F0

Рисунок 2.18 - Отрисовка объектов для визуализации

Описание формирования направляющих векторов для сенсоров, то есть в зависимости от положения объекта-пилота, располагать сенсоры в строго определенных точках (типа спереди и по бокам) и строго определенно направленных (типа вперед и влево-вправо) (рисунок 2.19).

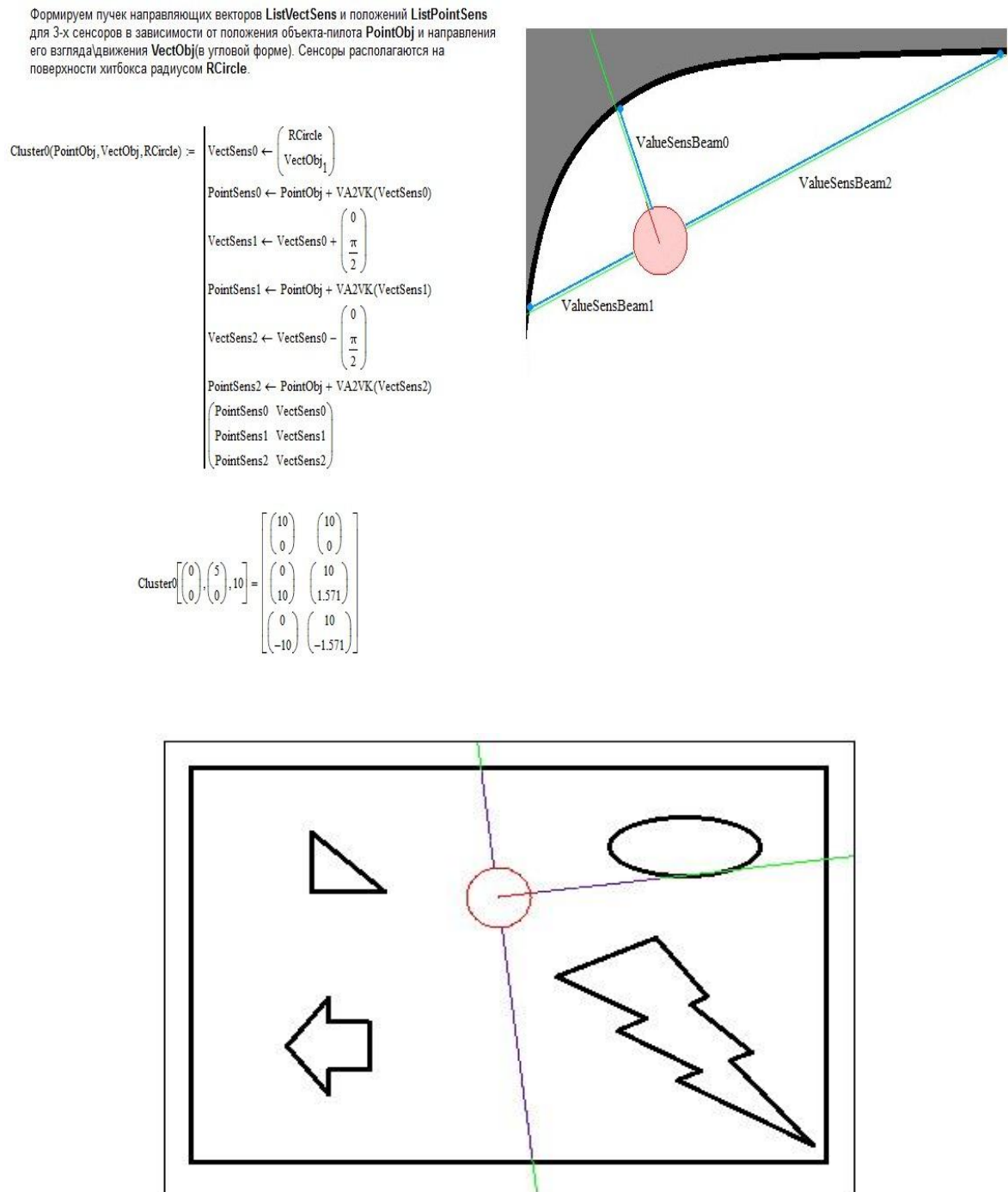


Рисунок 2.19 Описание формирования направляющих векторов для сенсоров

Вот так выглядит программа для вычисления положений сенсоров, значений сенсоров и рисующая всю сцену (рисунок 2.20).

```

TimeOne(RCircle, time, PointObj, VectObj, RacingWay) := "НИЖЕ РАСЧЕТНАЯ ЧАСТЬ"
FeeldKolliz ← RacingWay
FeeldKolliz ← DrowCircleKolliz(FeeldKolliz, PointObj0, PointObj1, RCircle, 0)
ListS ← Cluster0(PointObj, VectObj, RCircle)
(ListPointSens ListVectSens) ← (ListS<0> ListS<1>)
for i ∈ 0..rows(ListPointSens) - 1
    ListValueSensBeami ← SensBeam(ListPointSensi, ListVectSensi, FeeldKolliz)
"НИЖЕ ВИЗУАЛИЗАЦИЯ \ ОТРИСОВКА"
ffon ← augment[(1 - RacingWay)·255, (1 - RacingWay)·255, (1 - RacingWay)·255]
fon ← M2T(ffon)
for i ∈ 0..rows(ListValueSensBeam) - 1
    fon ← DrowSensBeam0 ListPointSensi, ListVectSensi, fon,  $\begin{pmatrix} 0 \\ 255 \\ 0 \end{pmatrix}$ 
    fon ← DrowSensBeam1 ListPointSensi, ListVectSensi, ListValueSensBeami, fon,  $\begin{pmatrix} 100 \\ 0 \\ 200 \end{pmatrix}$ 
fon ← DrowObj PointObj, VectObj, RCircle, fon,  $\begin{pmatrix} 255 \\ 0 \\ 0 \end{pmatrix}$ 
fone ← T2M(fon)
fone

```

$$E := \text{TimeOne} \left[20, \text{time}, \begin{pmatrix} 200 \\ 100 \end{pmatrix}, \begin{pmatrix} 20 \\ -\pi \\ \frac{1}{180} \cdot 7 \end{pmatrix}, \text{RacingWay} \right]$$

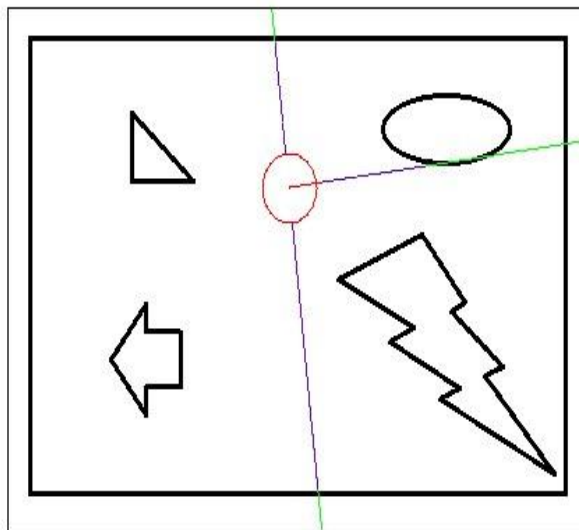


Рисунок 2.20 - программа для вычисления положений сенсоров

На рисунке 2.21 изображена схема одной итерация жизни объекта-пилота.

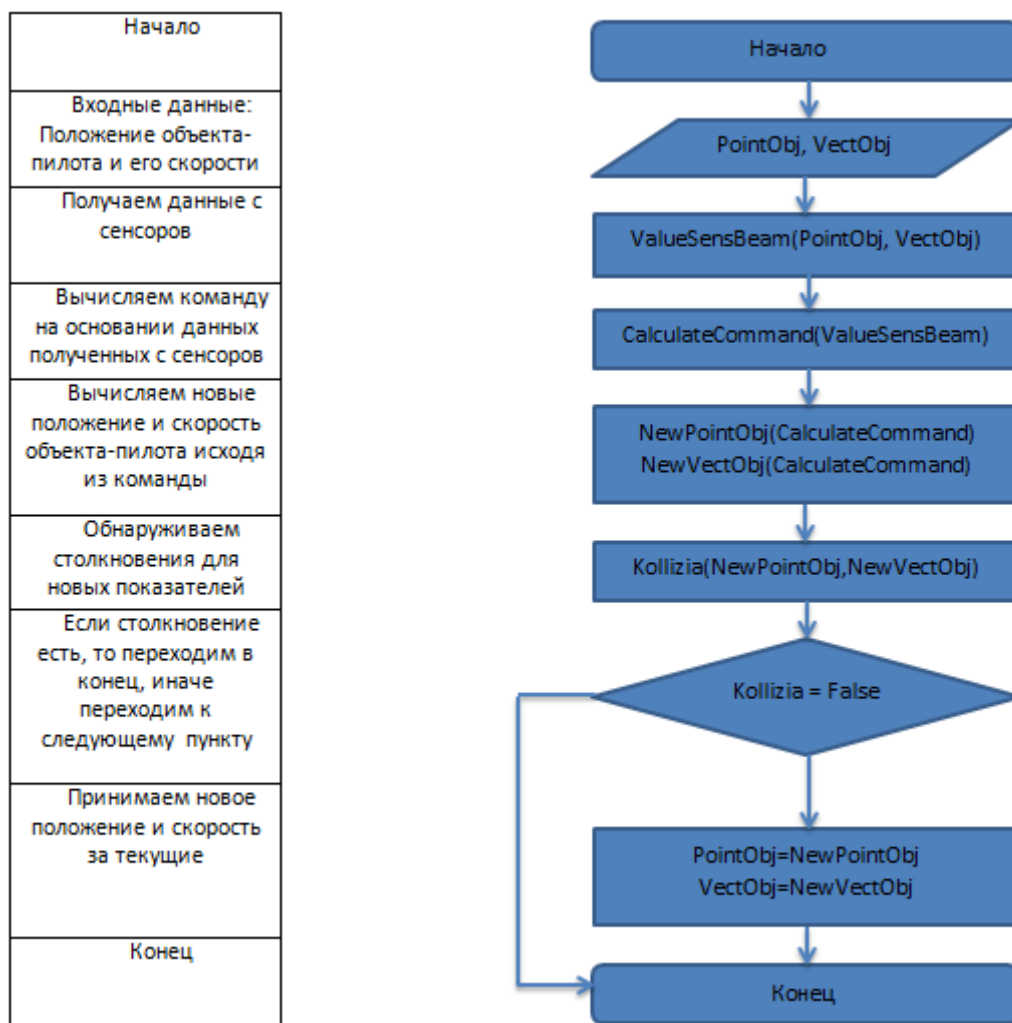


Рисунок 2.21 - Одна итерация жизни объекта-пилота

Исходными данными для препроцессора являются геометрическая модель объекта, чаще всего получаемая из подсистемы конструирования. Основная функция препроцессора - представление исследуемого объекта (детали) в сеточном виде, т.е. в виде множества конечных элементов.

Решатель - это программа, которая собирает модели отдельных конечных элементов в общую систему алгебраических уравнений и решает эту систему одним из методов разреженных матриц.

Постпроцессор служит для визуализации результатов решения в удобной для пользователя форме.

2.3 Описание библиотек

DrowLib - библиотека отрисовки данных представляет собой набор функций, отвечающих за формирование экрана, его фона, а также различных фигур отображающих процесс моделирования в вид пригодный для пользователя (в данном случае в двумерное изображение процесса моделирование движения объекта-пилота по "ландшафту").

Функция FonTens(X, Y, RGB) формирует экран размером X на Y пикселей и окрашивает фон цветом RGB, где RGB это вектор-столбец содержащий 3 элемента (R - величина красного цвета, G - величина зеленого цвета, B - величина синего цвета, каждое значение находится в интервале от 0 до 255 включительно) (рисунок 2.22).

$$\text{FonTens}(X, Y, \text{RGB}) := \left| \begin{array}{l} (r \ g \ b) \leftarrow (\text{RGB}_0 \ \text{RGB}_1 \ \text{RGB}_2) \\ \text{for } i \in 0..Y-1 \\ \quad \text{for } j \in 0..X-1 \\ \qquad T_{i,j} \leftarrow (r \ g \ b)^T \\ \qquad \qquad \qquad T \end{array} \right|$$

Рисунок 2.22 - Функция FonTens

Пример хранения информации о изображении в тензорном виде, изображение размером 5 пикселей на 5 пикселей, каждая ячейка матрицы является вектором из 3-х элементов, указывающих на значение в этой ячейке красного, зеленого и синего цвета соответственно (рисунок 2.23).

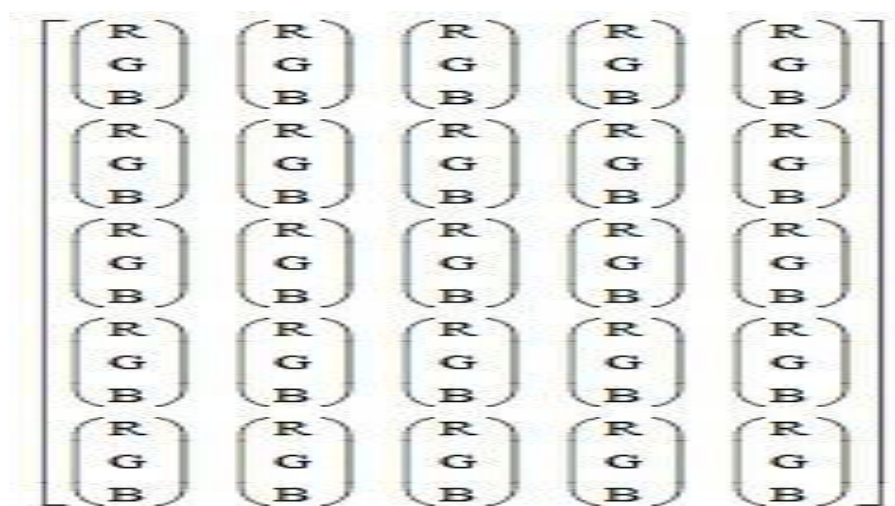


Рисунок 2.23 - Пример хранения информации в тензорном виде

Пример хранения информации о изображении в матричном виде, последовательно располагаются матрицы с значениями красного, зеленого и синего цветов изображения размером 5 пикселей на 5 пикселей (рисунок 2.24)



Рисунок 2.24 - Пример хранения информации в матричном виде

Функции позволяющие переводить тензорное представление в матричное T2M и матричное в тензорное M2T, это требуется для того что бы маткад мог адекватно выводить изображения (маткад работает с матричным

представлением, а в программе моделирования удобно работать с тензорным представлением) (рисунок 2.25).



Рисунок 2.25 - Функции позволяющие переводить тензорное представление в матричное

DrowLine позволяет рисовать на фоне Img какое-либо изображение в тензорном виде отрезок между точками с координатами (x1, y1) и (x2, y2) цветом RGB (рисунок 2.26).

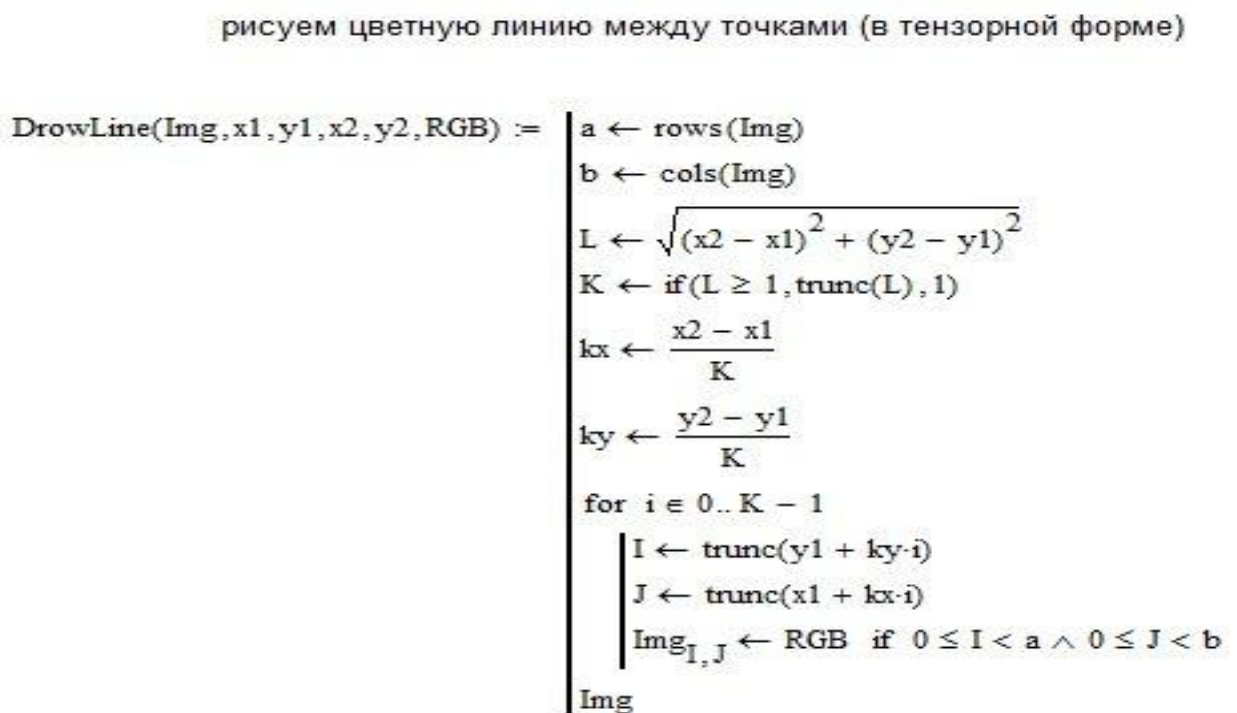


Рисунок 2.26 –Цветная линия в тензорной форме

Пример - рисуем отрезок на синем фоне F от точки (34, 15) до точки (55, 179), используем жёлтый цвет (255, 255, 0)(рисунок 2.27).

$$F1 := \text{DrawLines} \left[F, \begin{pmatrix} 34 & 55 \\ 15 & 179 \end{pmatrix}, \begin{pmatrix} 255 \\ 255 \\ 0 \end{pmatrix} \right]$$

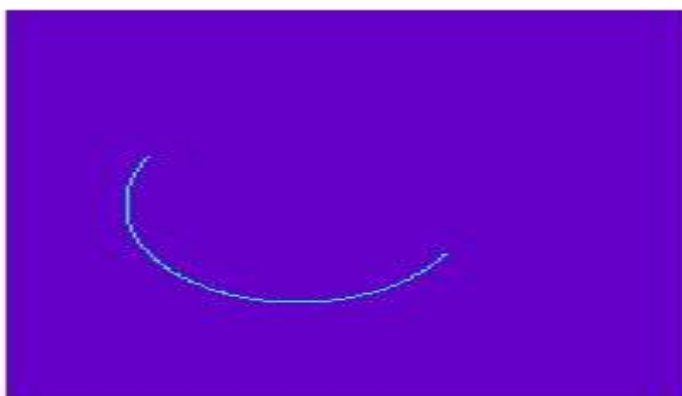


F0

Рисунок 2.27 – Пример рисунка

DrawSectio позволяет рисовать на фоне Img какое-либо изображение в тензорном виде, часть окружности с центром в точке (x0, y0) от угла alfa шириной угол beta, радиусом r и цветом RGB (рисунок 2.28).

$$F1 := \text{DrawSectio} \left[F, 85, 100, 50, \frac{\pi}{6}, \pi, \begin{pmatrix} 0 \\ 255 \\ 255 \end{pmatrix} \right]$$



F0

Рисунок 2.28 – Пример рисунка

Пример - рисуем часть окружности на фиолетовом фоне F с центром (85,100) от угла 30 (пи/6) градусов шириной 180 градусов полуокружность радиусом 50 пикселей, используем голубой цвет (0, 255, 255)(рисунок 2.29).

Рисуем часть окружности размером β

```

DrowSection(Img, x0, y0, r,  $\alpha$ ,  $\beta$ , RGB) :=
| a ← rows(Img)
| b ← cols(Img)
|  $L \leftarrow 2 \cdot 3.1415927 \cdot r \cdot \frac{\beta}{2\pi}$ 
|  $K \leftarrow \text{if}(L \geq 1, \text{trunc}(L), 1)$ 
|  $k\beta \leftarrow \frac{\beta}{K}$ 
| for i ∈ 0..K - 1
|   | I ← trunc(y0 + r · sin( $\alpha$  + i · k $\beta$ ))
|   | J ← trunc(x0 + r · cos( $\alpha$  + i · k $\beta$ ))
|   |  $\text{Img}_{I,J} \leftarrow \text{RGB}$  if  $0 \leq I < a \wedge 0 \leq J < b$ 
| Img

```

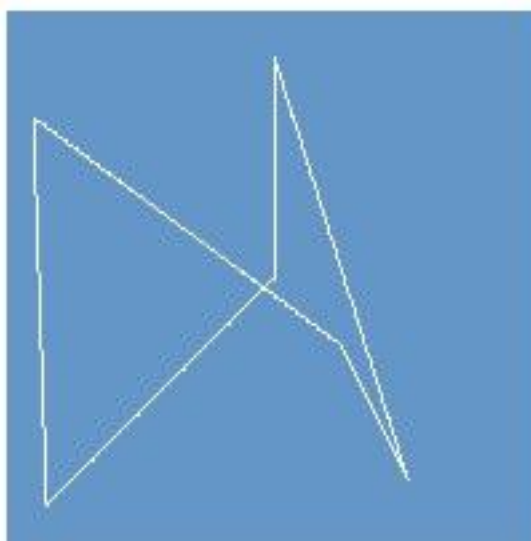
Рисунок 2.29—Часть окружности с центром в точке (x0, y0) от угла alpha шириной угол beta, радиусом r и цветом RGB;

DrowLines позволяет рисовать на фоне Img какое-либо изображение в тензорном виде множество отрезков между точками с координатами (x1, y1) и (x2, y2) заданных в горизонтальном массиве Lines цветом RGB.

Пример - рисуем отрезки на синем фоне F , используем белый цвет (255, 255, 255) (рисунок 2.30)

$$\text{Lines} := \begin{pmatrix} 10 & 15 & 100 & 100 & 150 & 125 & 10 \\ 40 & 185 & 100 & 17 & 175 & 125 & 40 \end{pmatrix}$$

$$\text{F1} := \text{DrawLines} \left[\text{F}, \begin{pmatrix} 10 & 15 & 100 & 100 & 150 & 125 & 10 \\ 40 & 185 & 100 & 17 & 175 & 125 & 40 \end{pmatrix}, \begin{pmatrix} 255 \\ 255 \\ 255 \end{pmatrix} \right]$$



F0

рисуем фигуру из непрерывного набора линий заданных началом и концом (в горизонтальной матрице)

$$\text{DrawLines}(\text{Img}, \text{Lines}, \text{RGB}) := \left| \begin{array}{l} a \leftarrow \text{cols}(\text{Lines}) \\ \text{for } i \in 0..a-2 \\ \quad \left(\begin{array}{cc} x1 & x2 \\ y1 & y2 \end{array} \right) \leftarrow \left(\begin{array}{cc} \text{Lines}_{0,i} & \text{Lines}_{0,i+1} \\ \text{Lines}_{1,i} & \text{Lines}_{1,i+1} \end{array} \right) \\ \quad \text{Img} \leftarrow \text{DrawLine}(\text{Img}, x1, y1, x2, y2, \text{RGB}) \\ \text{Img} \end{array} \right|$$

Рисунок 2.30 – Отрезки на синем фоне

DrawObj позволяет рисовать на фоне Img (какое-либо изображение в тензорном виде) объекта-пилота с координатами PointObj и направлением

VectObj заданного в угловом формате, размером RCircle цветом RGB. Пример - рисуем объекта-пилота на синем фоне F , используем белый цвет (255, 255, 255) (рисунок 2.31)

Рисуем объект (окружность со стрелкой/линией). VectObj задан в угловых координатах

```

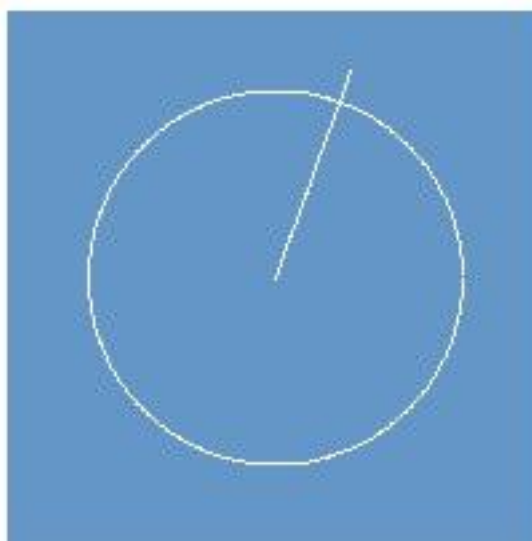
DrowObj(PointObj, VectObj, RCircle, Img, RGB) :=
  Img ← DrowSectio(Img, PointObj0, PointObj1, RCircle, 0, 2π, RGB)
  VectObj2 ← (RCircle·1.2 / VectObj1)
  Vect ← VA2VK(VectObj2)
  PointCur ← PointObj + Vect
  Img ← DrowLine(Img, PointObj0, PointObj1, PointCur0, PointCur1, RGB)
  Img

```

$$\text{PointObj} := \begin{pmatrix} x_0 \\ y_0 \end{pmatrix}$$

$$\text{VectObj} := \begin{pmatrix} \text{Long} \\ \text{Angle} \end{pmatrix}$$

$$F2 := \text{DrowObj} \left[\begin{pmatrix} 100.5 \\ 100 \end{pmatrix}, \begin{pmatrix} 3 \\ a \cdot \frac{\pi}{180} \end{pmatrix}, 70, F, \begin{pmatrix} 255 \\ 255 \\ 255 \end{pmatrix} \right]$$



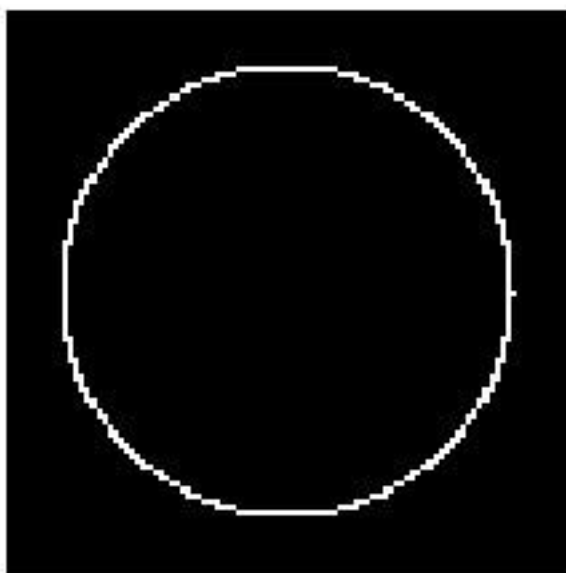
F0

Рисунок 2.31 - Объект-пилот на синем фоне

DrowSectioKolliz позволяет рисовать на фоне Img (Бинарное изображение) окружность с центром в точке (x0, y0) радиусом RCircle и цветом C (как правило C=1, так как изображение бинарное). Такое изображение используется в качестве круглого хитбокса для выявления коллизий.

Пример - рисуем часть окружности на черном фоне Binarize (поле заполненное нулями) с центром (50, 50) и радиусом 40 пикселей, C=1 (рисунок 2.32).

```
Bin := DrawCircleKolliz(Binarize,50,50,40,1)
```



Bin-255

```
DrawCircleKolliz(Img,x0,y0,RCircle,C) :=
|
a ← rows(Img)
b ← cols(Img)
L ← 2·3.1415927·RCircle
K ← if(L ≥ 1,2 trunc(L),1)
kβ ←  $\frac{2\pi}{K}$ 
for i ∈ 0..K - 1
|
| I ← trunc(y0 + RCircle·sin(i·kβ))
| J ← trunc(x0 + RCircle·cos(i·kβ))
| ImgI,J ← C if 0 ≤ I < a ∧ 0 ≤ J < b
|
Img
```

Рисунок 2.32 - Часть окружности на черном фоне Binarize

VectLib - библиотека работы с векторами представляет собой набор функций, отвечающих за формирование и работу с векторами, такую как

преобразование формы из угловой в координатную и обратно, вычисление длины вектора и наклона.

LongVect позволяет находить длину вектора заданного в координатной форме (рисунок 2.33).

Длина вектора

$$\text{LongVect}(\text{VK}) := \left| \begin{array}{l} x \leftarrow \text{VK}_0 \\ y \leftarrow \text{VK}_1 \\ L \leftarrow \sqrt{x^2 + y^2} \\ L \end{array} \right.$$

Рисунок 2.33 – Длина вектора

AngleVect позволяет находить угол наклона вектора относительно оси ОХ, заданного в координатной форме (рисунок 2.34)

угол наклона вектора с положительным направлением оси X

$$\text{AngleVect}(\text{VK}) := \left| \begin{array}{l} x \leftarrow \text{VK}_0 \\ y \leftarrow \text{VK}_1 \\ \text{ac} \leftarrow \arccos\left(\frac{x}{\sqrt{x^2 + y^2}}\right) \\ a \leftarrow \text{if}(y \geq 0, \text{ac}, 2\pi - \text{ac}) \\ a \end{array} \right.$$

Рисунок 2.34 – Угол наклона вектора с положительным направлением оси

Функции позволяющие переводить представление вектора из координатного в угловое VK2VA и из углового в координатное VA2VK.

Координатное представление отличается от углового тем, что элементами вектора являются величины его проекций по осям, в то время как при угловом представлении элементами вектора являются его длина и угол наклона с осью OX (рисунок 2.35).

переводим вектор из координатной формы в угловую

$$VK2VA(VK) := \begin{pmatrix} \text{LongVect}(VK) \\ \text{AngleVect}(VK) \end{pmatrix}$$

Переводим вектор из угловой формы в координатную

$$VA2VK(VA) := \begin{pmatrix} VA_0 \cdot \cos(VA_1) \\ VA_0 \cdot \sin(VA_1) \end{pmatrix}$$

Рисунок 2.35 - Перевод VK2VA и VA2VK

PhysicLib - библиотека имитации физики представляет собой набор функций, отвечающих за вычисление следующего положения на основе предыдущего под воздействием внешних сил.

Как таковой физики здесь нет, скорее это можно назвать реализацией первого и (частично) второго закона ньютона. То есть в отсутствии внешних сил тело сохраняет состояние покоя или равномерного прямолинейного движения. NextPoint позволяет вычислять следующее положение тела под воздействием внешних сил, зависит от положения тела PointObj, направления и скорости тела (да, здесь это одно и тоже) VectObj, вычисленного смещения, в следствии, вращения колёс (они куда-то тянут машинку, пытаюсь повернуть её и изменить скорость, а она сопротивляется) Calculate, периода времени time (дискретность времени влияющая на точность расчётов) и ограничения на максимальновозможные скорости перемещения и поворота (чтобы она не

разогналась до сверхсветовой и не вращалась на месте быстрее, чем пульсар при коллапсе) Далее будут описаны следующие вычисления положения в соответствии с текущими показателями положения PointObj (рисунок 2.36).

Вычисляем следующее положение в соответствии с текущими показателями положения PointObj, вектора движения VectObj (в угловой форме) и вычисленного, на основе работы сенсоров, смещения Calculate.

time - период вычисления нового положения \ дискретность времени (уменьшить чтобы увеличить точность). По умолчанию равно 0.1 секунды

ConditionForVector := $\begin{pmatrix} \text{"Нижняя граница скорости"} & \text{"Верхняя граница скорости"} \\ \text{"Нижняя граница поворота"} & \text{"Верхняя граница поворота"} \end{pmatrix}$

NextPoint(PointObj, VectObj, Calculate, time, ConditionForVector) :=

$$\begin{aligned} & \text{NewVectObj} \leftarrow \text{VectObj} + \text{Calculate} \\ & \text{NewVectObj} \leftarrow \begin{pmatrix} \text{ConditionForVector}_{0,0} \\ \text{NewVectObj}_1 \end{pmatrix} \quad \text{if } \text{NewVectObj}_0 < \text{ConditionForVector}_{0,0} \\ & \text{NewVectObj} \leftarrow \begin{pmatrix} \text{ConditionForVector}_{0,1} \\ \text{NewVectObj}_1 \end{pmatrix} \quad \text{if } \text{NewVectObj}_0 > \text{ConditionForVector}_{0,1} \\ & \text{NewVectObj} \leftarrow \begin{pmatrix} \text{NewVectObj}_0 \\ \text{ConditionForVector}_{1,0} + \text{VectObj}_1 \end{pmatrix} \quad \text{if } \text{Calculate}_1 < \text{ConditionForVector}_{1,0} \\ & \text{NewVectObj} \leftarrow \begin{pmatrix} \text{NewVectObj}_0 \\ \text{ConditionForVector}_{1,1} + \text{VectObj}_1 \end{pmatrix} \quad \text{if } \text{Calculate}_1 > \text{ConditionForVector}_{1,1} \\ & \text{NewVectObj2} \leftarrow \text{VA2VK}(\text{NewVectObj}) \\ & \text{NewPointObj} \leftarrow \text{PointObj} + \text{NewVectObj2} \cdot \text{time} \\ & \begin{pmatrix} \text{NewVectObj} \\ \text{NewPointObj} \end{pmatrix} \end{aligned}$$

Рисунок 2.36 – Вычисления положения

KollizLib - библиотека работы с коллизиями представляет собой набор функций, отвечающих за выявление и обработку коллизий, а также формирование "поля коллизий" и работу с ним.

Термин "Поле коллизий" не технический термин. Это некий артефакт используемого здесь метода определения коллизий, скорее обозначает проекцию моделируемого пространства в параллельное бинарное пространство, где известно только о том есть ли объект или его часть в указанной области и больше ничего, этого как раз и достаточно для определения коллизий.

FeeldKolliz позволяет формировать поле коллизий, размещая на нём хитбоксы объектов-пилотов, используя список объектов-пилотов ListObj (на случай если пилотов несколько) и файл поля (трассы) RacingWay.

Пример - формируем поле коллизий, используя нескольких объектов-пилотов (несколько машинок). Окружности - круглые хитбоксы этих объектов-пилотов, именно с ними и определяется соударение. Поле слева без объектов-пилотов, поле справа с объектами-пилотами (рисунок 2.37).

Формируем поле коллизий FeeldKolliz используя список объектов-пилотов ListObj (на случай если пилотов несколько) и файл поля/трассы RacingWay

```
FeeldKolliz(ListObj, RacingWay) :=
  c ← rows(ListObj)
  Feeld ← RacingWay
  for i ∈ 0..c-1
    x0 ← (ListObji,0)0
    y0 ← (ListObji,0)1
    RCircle ← ListObji,1
    Feeld ← DrawCircleKolliz(Feeld, x0, y0, RCircle, 1)
  Feeld
```

ListObj :=

PointObj0	RCircle0
PointObj1	RCircle1
PointObj2	RCircle2
PointObj3	RCircle3
PointObj4	RCircle4

PointObj :=

$$\begin{pmatrix} x0 \\ y0 \end{pmatrix}$$

$$\text{ListObj}_{0,0} := \begin{pmatrix} 20 \\ 20 \end{pmatrix} \quad \text{ListObj}_{0,1} := 10$$

$$\text{ListObj}_{1,0} := \begin{pmatrix} 40 \\ 60 \end{pmatrix} \quad \text{ListObj}_{1,1} := 30$$

$$\text{ListObj}_{2,0} := \begin{pmatrix} 75 \\ 40 \end{pmatrix} \quad \text{ListObj}_{2,1} := 20$$

$$D := \text{FeeldKolliz}(\text{ListObj}, \text{Feeld})$$

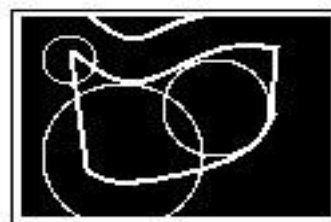


Рисунок 2.37 - Поле коллизий

DeleteObj позволяет удалять размещенные на поле коллизий хитбоксы объектов-пилотов, используя список объектов-пилотов ListObj (на случай если пилотов несколько) и индекс конкретного пилота N.

Пример - удаляем с поля коллизий хитбокс второго объекта-пилота.

Поле слева со всеми хитбоксами объектов-пилотов, поле справа с удаленным хитбоксом второго объекта-пилота (рисунок 2.38).

Удаляем соответствующий объект (не трассу) с поля коллизий

```

DeleteObj(FeeldKolliz,ListObj,N) :=
  x0 ← (ListObjN,0)0
  y0 ← (ListObjN,0)1
  RCircle ← ListObjN,1
  Feeld ← DrawCircleKolliz(FeeldKolliz,x0,y0,RCircle,0)
  Feeld

```

```

d := DeleteObj(D,ListObj,2)

```

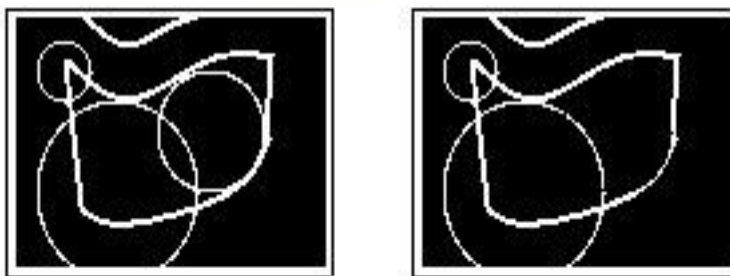


Рисунок 2.38 - Удаление хитбоксом второго объекта-пилота

KollizCircle позволяет определить наличие коллизии препятствия из FeeldKolliz с круглым хитбоксом радиусом RCircle в квадратной области близости размером RQuadrat для объекта-пилота находящегося в точке PointObj.

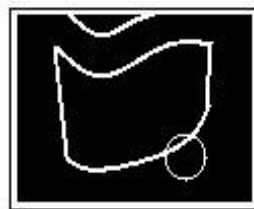
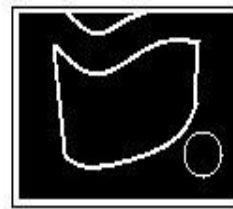
Пример – определяем коллизии для нескольких вариантов:

1 Объекта-пилота находящегося в точке (100, 65) на поле коллизий Feeld, с радиусом хитбокса RCircle=10 (тоесть диаметр =20) и стороной квадрата близости RQuadrat=25;

Так как хитбокс не касается препятствий - столкновений нет, то есть KollizCircle="false".

2Объекта-пилота находящегося в точке (85, 65) на поле коллизий Feeld, с радиусом хитбокса RCircle=10 (то есть диаметр =20) и стороной квадрата близости RQuadrat=25; Так хитбокс пересекает препятствие–столкновение (рисунок 2.39).

$$\text{KollizCircle}\left[\begin{pmatrix} 85 \\ 65 \end{pmatrix}, \text{Feeld}, 10, 25\right] = \text{"true"} \quad \text{KollizCircle}\left[\begin{pmatrix} 100 \\ 65 \end{pmatrix}, \text{Feeld}, 10, 25\right] = \text{"false"}$$



Проверяем наличие коллизии с круглым хитбоксом в квадратной области близости.
Условие: $R_{\text{Quadrat}} > 2 \cdot R_{\text{Circle}}$

```

KollizCircle(PointObj, FeeldKolliz, RCircle, RQuadrat) :=
  Rq ← ceil( (RQuadrat) / 2 )
  (a b) ← (rows(FeeldKolliz) cols(FeeldKolliz))
  (X Y) ← (trunc(PointObj0) trunc(PointObj1))
  Kollizia ← "false"
  for i ∈ -Rq..Rq
    for j ∈ -Rq..Rq
      I ← Y + i
      J ← X + j
      if 0 ≤ I < a ∧ 0 ≤ J < b
        LObj ← if (FeeldKollizI,J ≠ 0, √(i2 + j2), Rq)
        Kollizia ← "true" if LObj ≤ RCircle
        break if Kollizia = "true"
  Kollizia

```

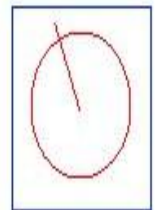


Рисунок 2.39 - Определение наличия коллизии препятствия

SensorLib - библиотека работы с сенсорами представляет собой набор функций, отвечающих за имитацию работы сенсоров, визуализацию их работы и размещение сенсоров относительно положения объекта-пилота.

SensBeam позволяет определить расстояние от сенсора находящегося в точке PointSens до препятствия расположенного в направлении вектора VectSens на поле коллизий FeeldKolliz (рисунок 2.40).

Сенсоры (лучи из точки **PointSens** в направлении вектора **VectSens** (в угловой форме) на поле коллизий **FeeldKolliz**). Возвращает расстояние до объекта или конца поля **FeeldKolliz**

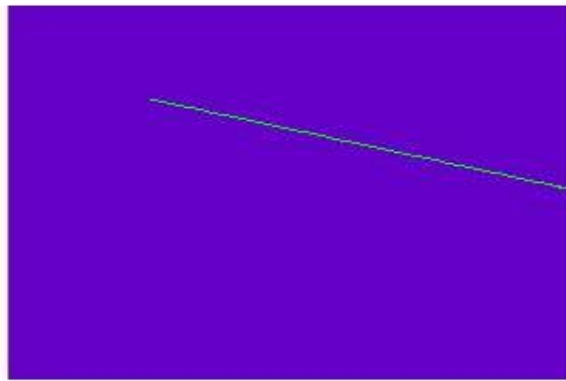
```
SensBeam(PointSens, VectSens, FeeldKolliz) :=
  a ← VectSens1
  (dx dy) ← (cos(a) sin(a))
  (X Y) ← (PointSens0 PointSens1)
  core ←  $\begin{pmatrix} 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 \end{pmatrix}^T$ 
  Q ← "Don't Stop"
  k ← 0
  while Q = "Don't Stop"
    (X Y) ← (X + dx Y + dy)
    (I J) ← (trunc(Y) trunc(X))
    for q ∈ 0..rows(core) - 1
      (i j) ← (I + coreq,0 J + coreq,1)
      Q ← "Stop" if i < 0 ∨ i ≥ rows(FeeldKolliz)
      Q ← "Stop" if j < 0 ∨ j ≥ cols(FeeldKolliz)
      Q ← "Stop" if FeeldKollizi,j ≠ 0 if Q ≠ "Stop"
    k ← k + 1
  VLong ←  $\begin{pmatrix} X - \text{PointSens}_0 & Y - \text{PointSens}_1 \end{pmatrix}^T$ 
  Long ← LongVect(VLong)
  Long
```

Рисунок 2.40 - Библиотека работы с сенсорами

DrowSensBeam0 позволяет визуализировать луч сенсора из точки PointSens в направлении VectSens цветом RGB на фоне Fon (в тензорном виде).

Пример - визуализируем луч сенсора из точки (50, 50) под углом 18 градусов (вектор (1, $\pi/10$)) зеленого цвета RGB=(0, 255, 0) на фоне фиолетового цвета (100, 0, 200) (рисунок 2.41).

$$F2 := \text{DrawSensBeam0} \left[\begin{pmatrix} 50 \\ 50 \end{pmatrix}, \begin{pmatrix} 1 \\ \frac{\pi}{10} \end{pmatrix}, F, \begin{pmatrix} 0 \\ 255 \\ 0 \end{pmatrix} \right]$$



F0

Рисуем луч цвета RGB из точки PointSens в направлении вектора VectSens (в угловой форме), до конца поля отрисовки Fon (в тензорном виде)

```
DrawSensBeam0(PointSens, VectSens, Fon, RGB) :=
| a ← rows(Fon)
| b ← cols(Fon)
| A ← VectSens1
| (x1 y1) ← (PointSens0 PointSens1)
| dia ← √(a2 + b2)
| (x2 y2) ← (x1 + dia·cos(A) y1 + dia·sin(A))
| Fon ← DrawLine(Fon, x1, y1, x2, y2, RGB)
| Fon
```

Рисунок 2.41 – Луч сенсора из точки PointSens в направлении VectSens

DrawSensBeam1 позволяет визуализировать луч сенсора длиной ValueSensBeam из точки PointSens в направлении VectSens цветом RGB на фоне Fon (какое-либо изображение в тензорном виде).

Пример - визуализируем луч сенсора длиной 100 пикселей из точки (50, 50) под углом 30 градусов (вектор (1, $\pi/6$)) голубого цвета RGB=(0, 255, 255) на фоне фиолетового цвета (100, 0, 200) (рисунок 2.42).

Рисуем луч цвета RGB из точки PointSens в направлении вектора VectSens (в угловой форме), до точки соприкосновения луча (на расстояние ValueSensBeam) с объектом или границей (в тензорном виде)

$$\text{DrowSensBeam1}(\text{PointSens}, \text{VectSens}, \text{ValueSensBeam}, \text{Fon}, \text{RGB}) := \left\{ \begin{array}{l} A \leftarrow \text{VectSens}_1 \\ (x1 \ y1) \leftarrow (\text{PointSens}_0 \ \text{PointSens}_1) \\ \text{VecA} \leftarrow (\text{ValueSensBeam} \ A)^T \\ \text{VecK} \leftarrow \text{VA2VK}(\text{VecA}) \\ (x2 \ y2) \leftarrow (x1 + \text{VecK}_0 \ y1 + \text{VecK}_1) \\ \text{Fon} \leftarrow \text{DrowLine}(\text{Fon}, x1, y1, x2, y2, \text{RGB}) \\ \text{Fon} \end{array} \right.$$

$$\text{F2} := \text{DrowSensBeam1} \left[\begin{pmatrix} 50 \\ 50 \end{pmatrix}, \begin{pmatrix} 1 \\ \frac{\pi}{6} \end{pmatrix}, 100, \text{F}, \begin{pmatrix} 0 \\ 255 \\ 255 \end{pmatrix} \right]$$


F0

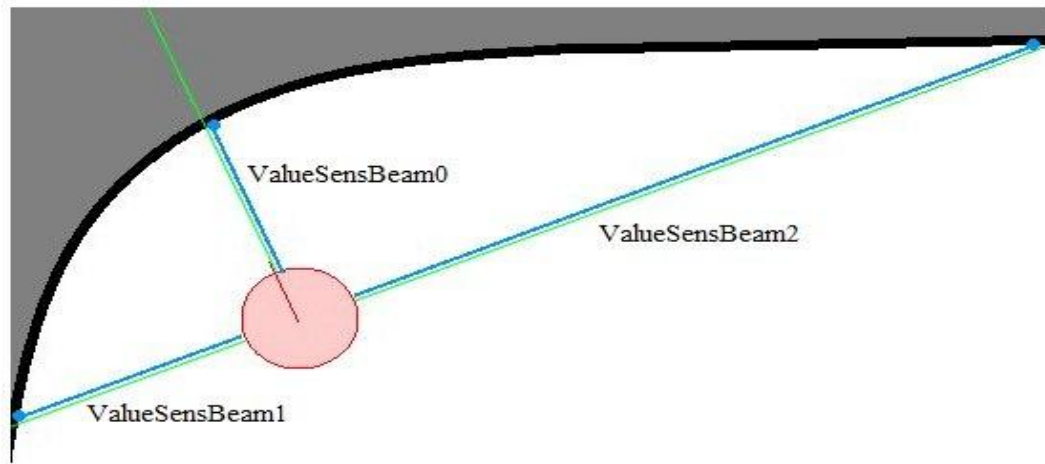
Рисунок 2.42 – Луч сенсора длиной ValueSensBeam из точки PointSens в направлении VectSens

Cluster0 позволяет формировать кластер из сенсоров относительно объекта-пилота находящегося в точке PointObj и ориентированного по вектору VectObj с радиусом хитбокса RCircle. Пример формируем кластер из 3х сенсоров (один спереди, один слева, один справа) для объекта-пилота находящегося в точке (0, 0) ориентированного в направлении вектора (5, 0).

То есть длиной 5 единиц и под углом 0 градусов) с радиусом хитбокса 10 единиц, во втором варианте аналогично, но вектор ориентации $(5, \pi/6)$, то есть под углом 30 градусов. Результаты отличаются, так как объект-пилот повернут на разные углы (рисунок 2.43).

Формируем пучек направляющих векторов **ListVectSens** (в угловой форме) и положений **ListPointSens** для 3-х сенсоров в зависимости от положения объекта-пилота **PointObj** и направления его взгляда/движения **VectObj** (в угловой форме). Сенсоры располагаются на поверхности хитбокса радиусом **RCircle**.

$$\text{Cluster0}(\text{PointObj}, \text{VectObj}, \text{RCircle}) := \left\{ \begin{array}{l} \text{VectSens0} \leftarrow \begin{pmatrix} \text{RCircle} + 1 \\ \text{VectObj}_1 \end{pmatrix} \\ \text{PointSens0} \leftarrow \text{PointObj} + \text{VA2VK}(\text{VectSens0}) \\ \text{VectSens1} \leftarrow \text{VectSens0} + \begin{pmatrix} 0 \\ \frac{\pi}{2} \end{pmatrix} \\ \text{PointSens1} \leftarrow \text{PointObj} + \text{VA2VK}(\text{VectSens1}) \\ \text{VectSens2} \leftarrow \text{VectSens0} - \begin{pmatrix} 0 \\ \frac{\pi}{2} \end{pmatrix} \\ \text{PointSens2} \leftarrow \text{PointObj} + \text{VA2VK}(\text{VectSens2}) \\ \begin{pmatrix} \text{PointSens0} & \text{VectSens0} \\ \text{PointSens1} & \text{VectSens1} \\ \text{PointSens2} & \text{VectSens2} \end{pmatrix} \end{array} \right.$$



$$\text{Cluster0} \left[\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 5 \\ 0 \end{pmatrix}, 10 \right] = \begin{bmatrix} \begin{pmatrix} 11 \\ 0 \end{pmatrix} & \begin{pmatrix} 11 \\ 0 \end{pmatrix} \\ \begin{pmatrix} 0 \\ 11 \end{pmatrix} & \begin{pmatrix} 11 \\ 1.571 \end{pmatrix} \\ \begin{pmatrix} 0 \\ -11 \end{pmatrix} & \begin{pmatrix} 11 \\ -1.571 \end{pmatrix} \end{bmatrix} \quad \text{Cluster0} \left[\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 5 \\ \frac{\pi}{6} \end{pmatrix}, 10 \right] = \begin{bmatrix} \begin{pmatrix} 9.526 \\ 5.5 \end{pmatrix} & \begin{pmatrix} 11 \\ 0.524 \end{pmatrix} \\ \begin{pmatrix} -5.5 \\ 9.526 \end{pmatrix} & \begin{pmatrix} 11 \\ 2.094 \end{pmatrix} \\ \begin{pmatrix} 5.5 \\ -9.526 \end{pmatrix} & \begin{pmatrix} 11 \\ -1.047 \end{pmatrix} \end{bmatrix}$$

Рисунок 2.43 - Кластер из сенсоров относительно объекта-пилота

CalculeLib - библиотека имитации вычислительного блока (бортового компьютера) представляет собой набор функций, отвечающих за формирование зависимостей для обработки сигналов полученных от сенсоров, распределения и структурирования списка сенсоров участвующих в процессе моделирования и получения итогового решения по поводу управления скоростью и направлением движения "автомобиля".

CommandSensor0 позволяет определить изменение скорости объекта-пилота в зависимости от показаний сенсора ValueSensBeam0 и условий (нижней и верхней границы работы сенсора) наложенных на сенсор ConditionForSensor0. Ниже конкретная реализация в рамках смоделированной задачи (рисунок 2.44).

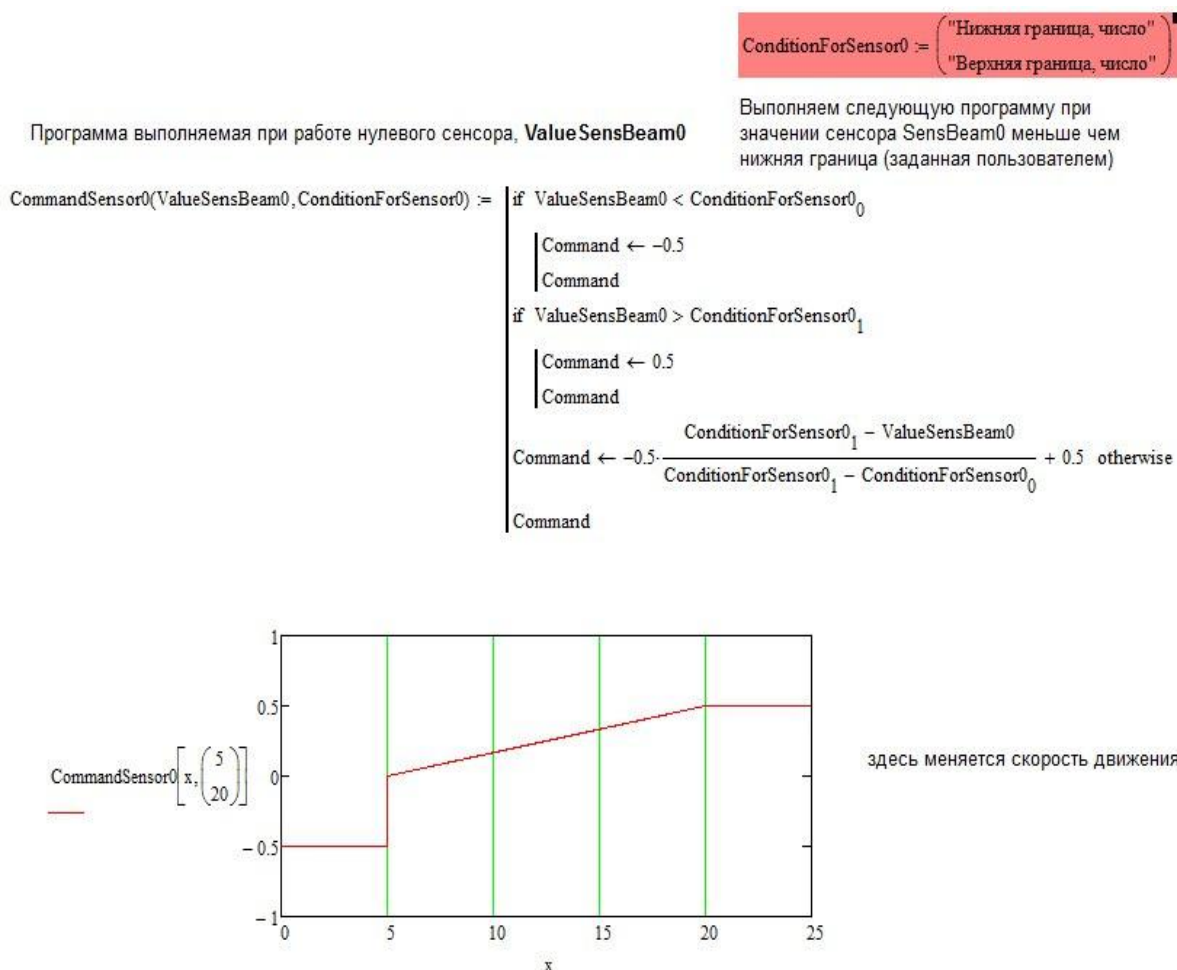


Рисунок 2.44 – Изменение скорости объекта-пилота в зависимости от показаний сенсора

CommandSensor1 позволяет определить изменение угла движения объекта-пилота в зависимости от показаний сенсора ValueSensBeam1 и условий (нижней и верхней границы работы сенсора) наложенных на сенсор ConditionForSensor1.

Ниже конкретная реализация в рамках смоделированной задачи.

```

CommandSensor1(ValueSensBeam1, ConditionForSensor1) :=
  if ValueSensBeam1 < ConditionForSensor10
  |
  | Command ←  $-\frac{\pi}{60}$ 
  | Command
  if ValueSensBeam1 > ConditionForSensor11
  |
  | Command ← 0
  | Command
  Command ←  $-\frac{\pi}{60} \cdot \frac{\text{ConditionForSensor1}_1 - \text{ValueSensBeam1}}{\text{ConditionForSensor1}_1 - \text{ConditionForSensor1}_0}$  otherwise
  | Command

```

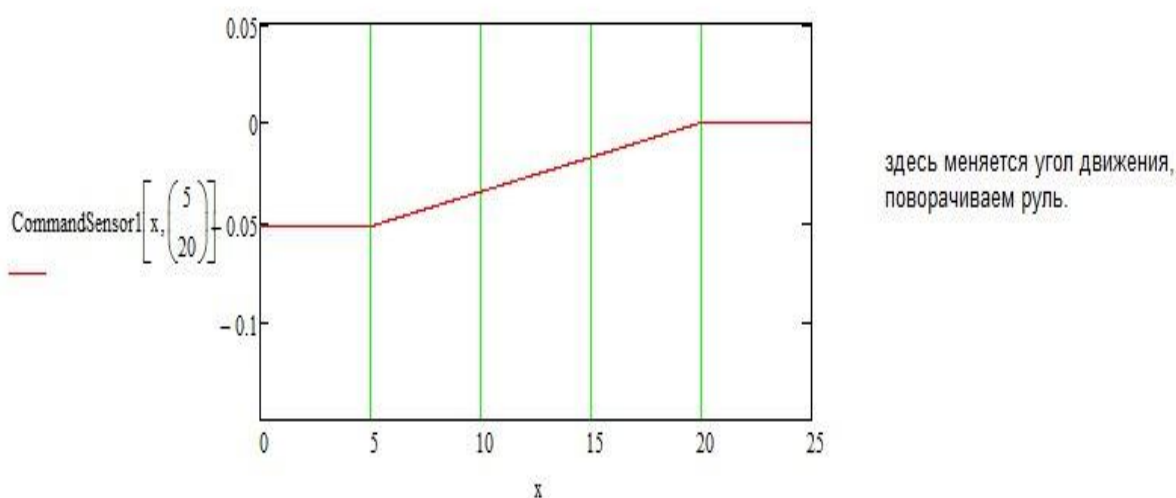


Рисунок 2.45 – Изменение угла движения объекта-пилота в зависимости от показаний сенсора

CommandSensor2 позволяет определить изменение угла движения объекта-пилота в зависимости от показаний сенсора ValueSensBeam2 и условий (нижней и верхней границы работы сенсора) наложенных на сенсор ConditionForSensor2.

Далее конкретная реализация в рамках смоделированной задачи (рисунок 2.46).

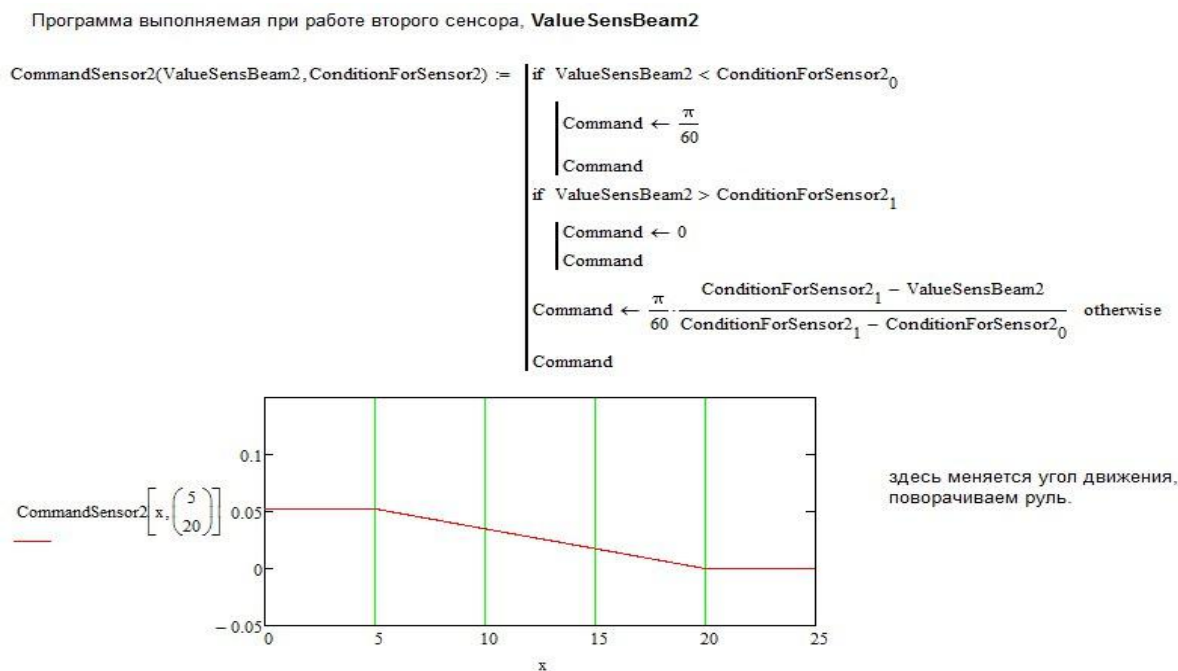


Рисунок 2.46 - Изменение угла движения объекта-пилота

Calculate - позволяет вычислить итоговое изменение скорости и угла движения объекта-пилота в зависимости от списка показаний сенсоров участвующих в процессе моделирования ValueListCalculate (рисунок 2.47).



Рисунок 2.47 - Вычисление изменения скорости и угла движение

Всё выделенное красным цветом подлежит обязательному заполнению руками, поэтому нужно делать препроцессор, где и будет заполняться такая информация, но в более наглядном виде, более понятным пользователю. Размещать сенсоры на модельке автомобиля, и отмечать галочками какие сенсоры будет в решении для этого и нужен препроцессор. После того как все библиотеки описаны остался сам файл решения Game03 просто входные данные (рисунок 2.48).

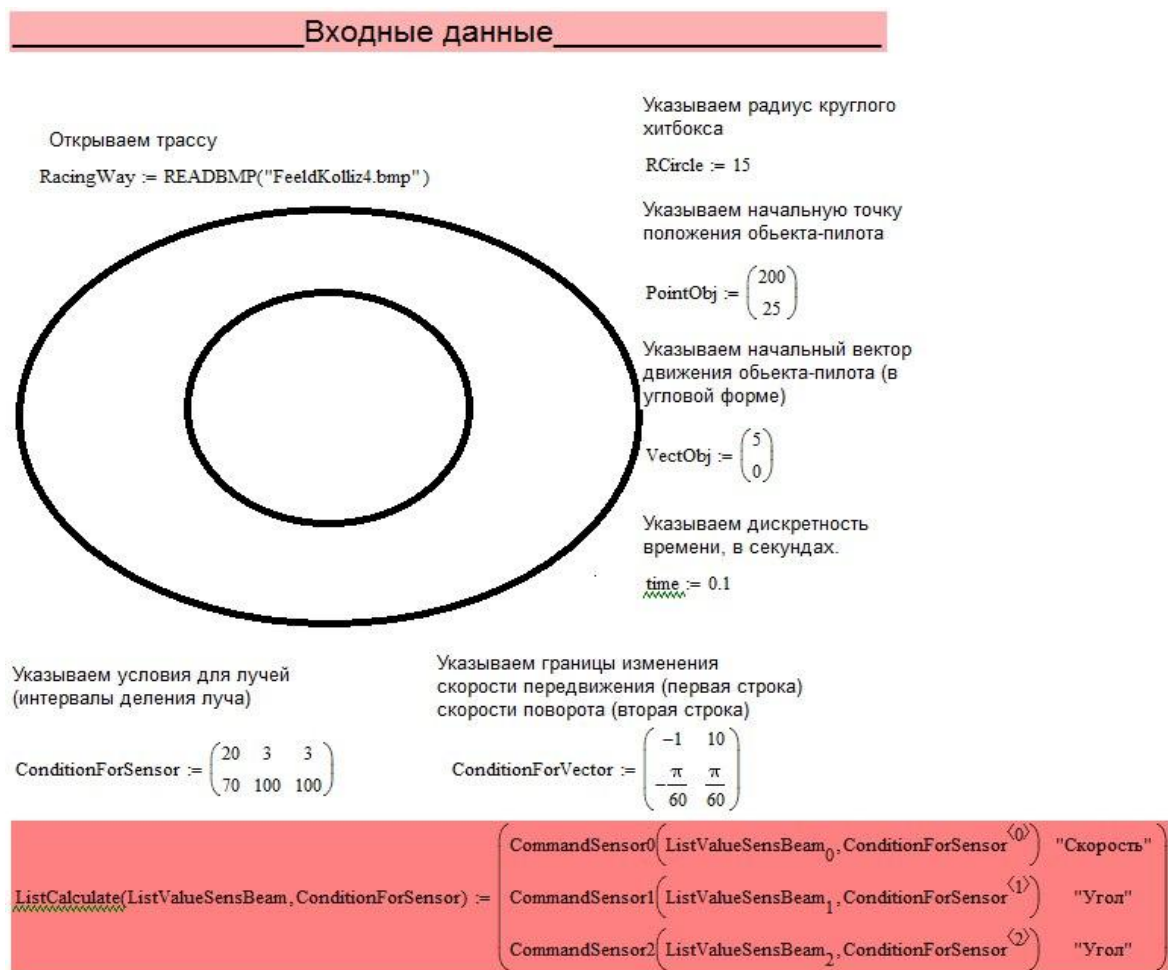


Рисунок 2.48 – Входные данные

TimeOne позволяет вычислить следующее положение объекта-пилота и следующую ориентацию объекта-пилота на трассе RacingWay по текущему положению PointObj и ориентации VectObj объекта-пилота. Таким образом, имея положение и ориентацию в момент времени t, эта функция вычисляет

положение и ориентацию, которая будет в момент времени $t+time$ ($time=0,1с$ как указано в входных данных) (рисунок 2.49).

Расчитываем одну итерацию цикла жизни одного объекта-пилота

```

TimeOne(RCircle, time, PointObj, VectObj, RacingWay) := "НИЖЕ РАСЧЕТНАЯ ЧАСТЬ"
RQuadrat ← 2(RCircle + 4)
FeeldKolliz ← RacingWay
ListS ← Cluster0(PointObj, VectObj, RCircle)
(ListPointSens ListVectSens) ← (ListS<0> ListS<1>)
for i ∈ 0..rows(ListPointSens) - 1
    ListValueSensBeami ← SensBeam(ListPointSensi, ListVectSensi, FeeldKolliz)
ValueListCalculate ← ListCalculate(ListValueSensBeam, ConditionForSensor)
Command ← Calculate(ValueListCalculate)
NewData ← NextPoint(PointObj, VectObj, Command, time, ConditionForVector)
(NewPointObj NewVectObj) ← (NewData1 NewData0)
Kollizia ← KollizCircle(NewPointObj, FeeldKolliz, RCircle, RQuadrat)
break if Kollizia = "true"
(PointObj VectObj) ← (NewPointObj NewVectObj) if Kollizia = "false"
(PointObj VectObj ListPointSens ListVectSens ListValueSensBeam Kollizia)T

```

Рисунок 2.49 - Вычисление положения и ориентации

Visual позволяет визуализировать дорогу, объекта-пилота и лучи сенсоров в момент времени t . Такая последовательная визуализация по моментам времени позволяет анимировать процесс моделирования автопилотирования (рисунок 2.50).

```

Visual(E, RCircle) := "НИЖЕ ВИЗУАЛИЗАЦИЯ \ ОТРИСОВКА"
(PointObj VectObj ListPointSens ListVectSens ListValueSensBeam) ← (E0 E1 E2 E3 E4)
ffon ← augment[(1 - RacingWay)·255, (1 - RacingWay)·255, (1 - RacingWay)·255]
fon ← M2T(ffon)
for i ∈ 0..rows(ListValueSensBeam) - 1
    fon ← DrawSensBeam0 [ ListPointSensi, ListVectSensi, fon,  $\begin{pmatrix} 0 \\ 255 \\ 0 \end{pmatrix}$  ]
    fon ← DrawSensBeam1 [ ListPointSensi, ListVectSensi, ListValueSensBeami, fon,  $\begin{pmatrix} 100 \\ 0 \\ 200 \end{pmatrix}$  ]
    fon ← DrawObj [ PointObj, VectObj, RCircle, fon,  $\begin{pmatrix} 255 \\ 0 \\ 0 \end{pmatrix}$  ]
fone ← T2M(fon)
fone

```

Рисунок 2.50 - Последовательная визуализация по моментам времени

Live – функция, содержащая главный цикл процесса моделирования, внутри которого происходит вычисление следующего положения и ориентации объекта и проверка на коллизии, а также визуализация нового вычисленного положения и ориентации объекта-пилота и лучей сенсора (рисунок 2.51).

Организуем цикл расчетов жизни объекта-пилота

```

Live(N) :=
  q ← 0
  for t ∈ 0..N
    D ← TimeOne(RCircle, time, PointObj, VectObj, RacingWay)
    q ← q + 1 if D0 - PointObj = 0 ∧ (D1)1 = VectObj1
    if q = 5
      D1 ← D1 +  $\begin{pmatrix} 0 \\ \frac{\pi}{60} \end{pmatrix}$ 
      q ← 0
    (PointObj VectObj) ← (D0 D1)
    Z ← Visual(D, RCircle)
    name ← concat("Data1\", num2str(t), ".jpg")
    WRITERGB(name, Z)
    break if D5 = "true"
  D

```

Рисунок 2.51 –Цикл расчетов жизни объекта пилота

Вывод

Была произведена разработка алгоритмов управления автономными транспортными средствами. Разработан вариант определения нахождения автомобиля в текущий момент времени. Использование нового термина коллизия, что означает обнаружение пересечений между собой двух или более объектов. Производился расчет коллизии. Приводят пример визуализации дороги, объекта-пилота и луча сенсоров в момент времени t. Описываются библиотеки используемые в работе.

3 Затраты на реализацию проекта

3.1 Расчет предпроизводственных затрат

Расчет затрат на проектирование разработки проводится в следующей последовательности:

Определяется полный перечень работ;

Определяется трудоемкость работ;

Распределяются работы между исполнителями;

Определяется длительность цикла выполнения этапа;

Определяются затраты на проектирование, т.е. составляется смета.

Полный перечень работ дается по этапам:

- научные исследования, разработка технического задания, технического предложения (проведение теоретических и экспериментальных исследований);
- проектно-конструкторские работы (эскизный проект; технический проект; разработка рабочей документации); изготовление опытного образца (макета);
- технологическая подготовка производства (обеспечение технологичности конструкции); разработка технологических маршрутов и процессов; расчеты трудоемкости операций и материалоемкости детали; конструирование оснастки, специального и вспомогательного оборудования и их изготовление; отладка и освоение изделия.

Определение трудоемкости работ.

Трудоемкость работ складывается из трудоемкости разработки чертежей, спецификаций и другой технической документации:

$$T_p = T_q + T_{т.д.} + T_{т.м.} + T_{п} + T_{н.к.}, \quad (3.1)$$

где: T_q - трудоемкость разработки чертежей, ч;

$T_{т.д.}$ - трудоемкость выполнения текстовых документов, ч;

$T_{т.м.}$ - трудоемкость выполнения текстовых материалов, ч;

$T_{п}$ - трудоемкость проверки технической документации, ч;

$T_{н.к.}$ - трудоемкость норм контроля, ч.

Трудоемкость разработки чертежей. В данном случае группа новизны В - проектирование сборочных единиц, схем и деталей изделий с введением значительных технических и принципиальных изменений составных частей с новыми параметрами, требующих экспериментальной проверки.

Группа сложности при конструировании деталей III - сложные детали, требующие специальные конструкторские расчеты, геометрические формы их представляют собой сочетание простых и сложных поверхностей

Чертежи общего вида - 30 чел./ч. - 5 листа;

Теоретический чертеж - 8 чел./ч. - 4 листа;

Чертежи электрических принципиальных схем - 85 чел./ч. - 1 лист.

$$T_{\text{ч}} = (5 \times 30) + (4 \times 8) + (1 \times 85) = 267 \text{ (ч)}$$

Трудоемкость выполнения текстовых документов определяется по нормативам в зависимости от вида документа:

$$T_{\text{т.д.}} = \sum_{i=1}^m (T_{\text{сп}} + T_{\text{пок}} + T_{\text{п.ч.}}) \times i + T_{\text{т.о.}} + T_{\text{инст}} + T_{\text{т.у.}} + T_{\text{п.и.}}, \quad (3.2)$$

где: $T_{\text{сп.}}$ - трудоемкость разработки спецификации, чел./ч.;

$T_{\text{пок.}}$ - трудоемкость составления ведомости покупных изделий, чел./ч.;

$T_{\text{п.ч.}}$ - трудоемкость составления перечня чертежей, чел./ч.;

$T_{\text{т.о.}}$ - трудоемкость составления технического описания, чел./ч.;

$T_{\text{инст.}}$ - трудоемкость разработки инструкции по эксплуатации и монтажу, чел./ч.;

$T_{\text{т.у.}}$ - трудоемкость разработки технических условий, чел./ч.;

$T_{\text{п.и.}}$ - трудоемкость разработки программы испытаний, чел./ч.;

m - количество агрегатов (сборок) в изделии.

В данном проекте:

$$T_{\text{сп1}} = 5 \text{ ч.}, T_{\text{сп2}} = 5 \text{ ч.}, T_{\text{пок1}} = 10 \text{ ч.}, T_{\text{пок2}} = 10 \text{ ч.}, T_{\text{п.ч1}} = 5 \text{ ч.}, T_{\text{п.ч2}} = 5 \text{ ч.},$$

$T_{\text{т.о.}} = 15 \text{ ч.}, T_{\text{инст}} = 15 \text{ ч.}, m = 2$, тогда:

$$T_{\text{т.д.}} = (5 + 5 + 10 + 10 + 5 + 5) \times 2 + 15 + 15 = 110 \text{ (чел./ч)}$$

Нормы времени на выполнение текстовых материалов зависят от вида работ

$$T_{\text{т.м.}} = \sum_{i=1}^m P_i \times T_{\text{вр}i}, \quad (3.3)$$

где P_i - количество текстовых материалов на каждый вид выполняемых работ;

$T_{\text{вр}i}$ - норма времени на вид выполненной работы, ч.;

m - количество видов выполняемой работы ($m = 7$).

В таблице 3.1 приведены нормы времени на выполнение текстовых материалов.

Таблица 3.1 – Норма времени на выполнение текстовых материалов.

Вид работы	Норма времени, ч.	Количество текстовых материалов
обзор литературы	25	5
расчет по формулам без таблиц	20	1
расчет с использованием таблиц	20	1
табличный расчет по исходным данным из чертежей	20	1
таблицы исходных данных	16	1
вычерчивание схем и эскизов	20	3
оформление начисто	14	1

$$T_{\text{т.м.}} = (25 \times 5) + 20 + 20 + 20 + 16 + (20 \times 3) + 14 = 275, \text{ ч.}$$

На проверку технической документации время предусматривается дополнительно в зависимости от характера работ:

$$T_{\text{п}} = \sum_{i=1}^m T_{\text{д}i} \times q_{\text{д}i}, \quad (3.4)$$

где: $T_{\text{д}i}$ - трудоемкость проверяемого вида технической документации, (чертежи, текстовые документы, текстовые материалы),

D_i - норматив на проверку отдельного вида технической документации равный 2%.

$$T_{\text{ч}} = 267 \text{ ч}$$

$$T_{\text{т.д.}} = 110 \text{ ч}$$

$$T_{\text{т.м.}} = 275 \text{ ч}$$

$$T_{дi} = T_{ч} + T_{т.д.} + T_{т.м.}, \quad (3.5)$$

$$T_{дi} = 267 + 110 + 275 = 652 \text{ (ч)}$$

$$T_{п} = 652 \times 0,02 = 13,04 \text{ (ч)}$$

Время на выполнение норм контроля.

$$T_{н.к.} = T_{р} \times q_{н.к.}, \quad (3.6)$$

$$T_{н.к.} = (267 + 110 + 275 + 13,04) \times 0,05 = 33,25 \text{ (чел./ч)}$$

Н.К - норматив норм контроля, 5 - 10%

Трудоемкость работ:

$$T_{р} = 267 + 110 + 275 + 13,04 + 33,25 = 698,29 \text{ (ч)}$$

Общая численность работников.

Определяется на основе рассчитанной трудоемкости работ:

$$\chi_{раб} = \frac{T_{раб}}{\Phi_{эф} \times K_{в}}, \quad (3.7)$$

где $T_{раб}$ - трудоемкость работ;

$\Phi_{эф}$ - эффективный фонд времени одного работника, ч.;

$K_{в}$ - коэффициент выполнения норм (на стадии проектирования принимается равным 1)

$$\chi_{раб} = \frac{698,29}{648 \times 1} \approx 1 \text{ (чел.)}$$

$$\Phi_{эф} = D_{р} \times q \times n, \quad (3.8)$$

$$\Phi_{эф} = 90 \times 8 \times 0,9 = 648 \text{ (ч)}$$

где $D_{р}$ - количество рабочих дней;

n - коэффициент плановых потерь времени (= 0,9);

q - продолжительность рабочего дня.

Распределение работы между исполнителями осуществляется с учетом их квалификации и длительности выполнения работы.

Длительность выполнения каждой стадии, этапа.

$$T_{с,э,рi} = \frac{T_i \times K_{дi} \times K_{к}}{\chi_i \times q \times K_{в}}, \quad (3.9)$$

$$T_{с,э,рi} = \frac{267 \times 1,1 \times 0,3}{1 \times 8 \times 1} = 11 \text{ ч}$$

где, T_i - трудоемкость стадии, этапа, работы, чел./ч.;

$Ч_i$ - численность работников, одновременно выполняющих данную стадию, этап, работу, чел.;

$Д_i$ - коэффициент, учитывающий дополнительное время на согласование, внесение изменений в техническую документацию и другие работы, не предусмотренные нормативами (1,1 - 1,5);

K_k - коэффициент перевода рабочих дней в календарные:

$$K_k = \frac{D_k}{D_p}, \quad (3.10)$$
$$K_k = \frac{30}{90} = 0,3$$

где, D_k - число календарных дней в плановом периоде.

Смета на проектирование

Составляется по следующим статьям затрат:

- 1 Расчёт затрат на конструкторские разработки.
- 2 Дополнительная заработная плата.
- 3 Отчисления на социальные нужды.
- 4 Затраты на машинное время.
- 5 Накладные расходы.
- 6 Затраты на создание опытного образца.

3.2 Смета на проектирование

При создании блока затраты на конструкторские разработки будут равняться затратам на оплату труда исполнителям, занимающимися созданием конструкторской документации.

Основная заработная плата.

$$З_{осн} = \sum_{i=1}^m Ч_i \times \Phi_i \times C_i, \quad (3.11)$$

где, $Ч_i$ - численность исполнителей, чел.;

Φ_i - фонд рабочего времени исполнителей, час;

C_i - тарифная ставка, руб./час.;

m- количество групп исполнителей.

Затраты на конструкторские разработки составляют:

$$З_{\text{констр}} = Ч_{\text{раб}} \times \Phi_{\text{эф}} \times C_{\text{ср}}, \quad (3.12)$$

$$З_{\text{констр}} = 1 \times 144 \times 23 = 3312 \text{ (руб)}$$

Дополнительная заработная плата:

$$З_{\text{доп}} = З_{\text{осн}} \times \frac{q_{\text{доп}}}{100}, \quad (3.13)$$

где $q_{\text{доп}}$ - норматив дополнительной заработной платы (30-200%).

$$З_{\text{доп}} = 3312 \times 0,3 = 993,6 \text{ (руб.)}$$

Отчисления на социальные нужды:

$$З_{\text{отч}} = (З_{\text{доп}} + З_{\text{осн}}) \times \frac{q_{\text{отч}}}{100}, \quad (3.14)$$

где $q_{\text{доп}}$ - норматив дополнительной заработной платы (34%).

$$З_{\text{отч}} = (993,6 + 3312) \times 0,34 = 1463,9 \text{ (руб.)}$$

Затраты на машинное время:

$$З_{\text{м.вр}} = T_{\text{м.вп}} \times Ц_{\text{м.вр}}, \quad (3.15)$$

где $T_{\text{м.вп}}$ - время выполнения работ на ЭВМ, час.;

$Ц_{\text{м.вр}}$ - цена одного часа машинного времени, руб.

$$З_{\text{м.вр}} = 40 \times 100 = 4000 \text{ (руб.)}$$

Затраты на создание опытного образца (макета)

- затраты на основные материалы;
- затраты на покупные комплектующие изделия и беспилотники;
- затраты на основную заработную плату производственных рабочих;
- затраты на дополнительную заработную плату производственных рабочих;
- затраты на отчисления с основной и дополнительной заработной платы производственных рабочих на социальные нужды;
- затраты по содержанию и эксплуатации оборудования;
- накладные расходы (цеховые и административные).

Затраты на основные материалы.

$$З_{\text{м}} = \sum_{i=1}^m (K_{\text{д}} \times H_{\text{м.д.}} \times Ц_{\text{м}}), \quad (3.16)$$

где, K_d - количество деталей (применяемость) на изделие, шт.;

$H_{м.д.}$ - норма расхода материала на деталь, кг, м;

C_m - цена за единицу материала, руб.;

m - количество наименований деталей.

Расчеты оформляются в таблицу 3.2:

Таблица 3.2 – Затраты на основные материалы

№	Наименование материала	Расход материала, кг.	Цена за единицу, руб.	Общая сумма затрат, руб.
1	Стеклотекстолит фольгированный СФ2-35-1.5,	0,052	300	15.6
2	Припой ПОС-61	0,016	500	8
3	Флюс ФКС	0.0085	87	0.74
4	Провод МГШВ-1	0.52 м	2.50	1.3
5	Лакокрасочные материалы	1	55	55
6	Ники шелковые №13	0.003	10	0.03
7	Трубка термоусадочная	0.2м	200	40
	ИТОГО			120,67

Материальные затраты (основные материалы, покупные комплектующие изделия и беспилотники) определяются с учетом транспортно - заготовительных расходов, которые составляют 3 - 4% от их стоимости.

Поэтому,

$$З_m = 120,67 + (120,67 \times 0,03) = 124,29 \text{ руб.}$$

Затраты на покупные комплектующие изделия и беспилотники.

Затраты на покупные комплектующие изделия и беспилотники рассчитываются по формуле :

$$З_{\text{пок}} = \sum_{i=1}^m K_{\text{бл}} \times K_{\text{поки}} \times C_{\text{поки}}, \quad (3.17)$$

где, $K_{\text{бл.}}$ - количество блоков в изделии;

$K_{\text{покл}}$ - количество покупных изделий и беспилотников, шт;

$C_{\text{покл}}$ - цена за единицу покупных изделий и беспилотников, руб.;

m - количество наименований покупных изделий и беспилотников.

Затраты на покупные комплектующие изделия и полуфабрикаты рассчитываются с учетом транспортно - заготовительных расходов, которые составляют 3 - 4% от их стоимости. Данные заносятся в таблицу 3.3.

Таблица 3.3 – Затраты на покупные комплектующие изделия и беспилотники

№	Наименование покупных изделий и беспилотников	Расход покупных изделий , шт	Цена за единицу, руб.	Общая сумма затрат, руб.
1	Держатель светодиода RTC-51	1	4	4,00
2	Индикатор АЛ307А	1	4	4,00
3	Диод КД102А	3	4	12,00
4	Стабилизатор 7805	1	7	7,00
5	Индуктивность AL0307	1	4	4,00
6	Конденсатор К10-17А	12	5	60,00
7	Конденсатор К53-18	2	15	30,00
8	Конденсатор К10-17Б	2	10	20,00
9	Микроконтроллер Z86E0208PSC	1	50	60,00
10	Резонатор НС-49S	1	10	10,00
11	Предохранитель стеклянный	1	5	5,00
12	Пьезоэл. излучатель SCS-17P-10	1	80	80,00
13	Радиатор FK301	1	10	10,00
14	Разъемы СР-50	2	50	100,00
15	Разъем РГ1Н-1-1	1	20	20,00
16	Резистор С2-23	19	2	38,00
17	Резистор СПЗ-39А	1	23	23,00
18	Транзистор КТ3102Е	2	20	40,00
19	Транзистор КТ3107Ж	1	10	10,00
20	Переключатель В100 G	1	10	10,00
21	Держатель предохранителя FH-301	1	20	20,00
	ИТОГО			567,00

Поэтому

$$З_{\text{пок}} = 567 + (567 \times 0,03) = 584,01 \text{ (руб.)}$$

Затраты на основную заработную плату производственных рабочих.

$$З_{\text{осн}} = \sum_{i=1}^m T_i \times C_{pi}, \quad (3.18)$$

$$C_{pi} = \frac{З_{\text{min}}}{K_{\text{ч}}} \times C_p = C_{1p} \times T_k, \quad (3.19)$$

где, T_i - трудоемкость изготовления изделия, нормо-часы

$$T_i = \sum_{i=1}^m t_i, \quad (3.20)$$

t_i — трудоемкость выполнения соответствующей операции;

C_{1p} — часовая ставка 1-го разряда работ;

$З_{\text{min}}$ — минимальная тарифная ставка;

C_{pi} — часовая тарифная ставка соответствующего разряда работ, руб;

T_k — тарифный коэффициент.

Используемые виды работ и их разрядность.

Заготовительные — 2 разряд.

Травление - 3 разряд.

Пайка — 4 разряд.

Сборочные — 3 разряд.

Наладочные — 5 разряд.

Часовая ставка первого разряда равняется 12,5 руб.

Тарифный коэффициент — T_k , зависит от разряда, так 2-му разряду соответствует $T_k = 1,3$; 3-ему — $T_k = 1,69$; 4-му — $T_k = 1,91$; 5-му — $T_k = 2,16$.

Таким образом:

$$C_{2p} = 1,3 \times 12,5 = 16,25 \text{ рублей};$$

$$C_{3p} = 1,69 \times 12,5 = 21,125 \text{ рублей};$$

$$C_{4p} = 1,91 \times 12,5 = 23,875 \text{ рублей};$$

$$C_{5p} = 2,16 \times 12,5 = 27 \text{ рублей}.$$

По полученным данным составим таблицу 3.4.

Таблица 3.4 – Расчет основной заработной платы рабочих

Виды работ	Трудоемкость работ, н.ч.	Разряд работ	Срi, руб.	Основная зарплата рабочих, руб
Заготовительные	1	2	16.25	16.25
Травление	4	3	21.125	84.5
Пайка	12	4	23.875	286.5
Сборочные	6	3	21.125	126.75
Наладочные	3	5	27	81

$$З_{\text{осн}} = 16,25 + 84,5 + 286,5 + 126,75 + 81 = 595 \text{ (руб.)}$$

Затраты на дополнительную заработную плату производственных работ.

$$З_{\text{доп}} = З_{\text{осн}} \times \frac{q_{\text{доп}}}{100}, \quad (3.21)$$

где $q_{\text{доп}}$ - норматив дополнительной заработной платы (30%).

$$З_{\text{доп}} = 595 \times 0,3 = 178,5 \text{ (руб.)}$$

Отчисления с основной и дополнительной заработной платы производственных рабочих на социальные нужды.:

$$З_{\text{отч}} = (З_{\text{осн}} + З_{\text{доп}}) \times \frac{q_{\text{отч}}}{100}, \quad (3.22)$$

Где $q_{\text{отч}}$ - отчисления на социальное страхование (26%).

$$З_{\text{отч}} = (595 + 178,5) \times 0,26 = 618,8 \text{ (руб.)}$$

Затраты на содержание и эксплуатацию оборудования:

$$З_{\text{с.э.о.}} = З_{\text{осн}} \times \frac{q_{\text{с.э.о.}}}{100}, \quad (3.23)$$

где $q_{\text{с.э.о.}}$ - норматив расходов на содержание и эксплуатацию оборудования, $q_{\text{с.э.о.}} = 65\%$

$$З_{\text{с.э.о.}} = 595 \times 0,65 = 386,75 \text{ руб.}$$

Цеховые расходы:

$$З_{\text{ц}} = (З_{\text{осн}} + З_{\text{с.э.о.}}) \times \frac{q_{\text{ц}}}{100}, \quad (3.24)$$

где $q_{ц}$ - норматив цеховых расходов (30%).

$$З_{ц} = (595 + 386,75) \times 0,3 = 294,5 \text{ руб.}$$

Общезаводские расходы.

$$З_{оз} = (З_{осн} + З_{с.э.о.}) \times \frac{q_{оз}}{100}, \quad (3.25)$$

где $q_{оз}$ - норматив общезаводских расходов (30%).

$$З_{оз} = (595 + 386,75) \times 0,35 = 343,6 \text{ (руб.)}$$

Производственная себестоимость определяется.

$$C_{пр} = З_{м} + З_{пок} + З_{осн} + З_{доп} + З_{отч} + З_{с.э.о.} + З_{ц} + З_{оз}, \quad (3.26)$$

$$\begin{aligned} C_{пр} &= 124,29 + 584,01 + 595 + 178,5 + 618,8 + 386,75 + 294,5 + 343,6 \\ &= 3125,45 \text{ (руб.)} \end{aligned}$$

Коммерческие расходы.

$$З_{к} = C_{пр} \times \frac{q_{к}}{100}, \quad (3.27)$$

где, $q_{к}$ - норматив внепроизводственных расходов (2%).

$$З_{к} = 3125,45 \times 0,02 = 62,5 \text{ (руб.)}$$

Полная себестоимость.

$$C_{пол} = C_{пр} + З_{к}, \quad (3.28)$$

$$C_{пол} = 3125,45 + 62,5 = 3187,95 \text{ (руб.)}$$

Расчёт затрат на проектирование и создание опытного образца.

$$З_{пр} = З_{констр} + З_{доп} + З_{отч} + З_{м.впр.} + C_{пол}, \quad (3.29)$$

$$З_{пр} = 3312 + 993,6 + 1463,9 + 4000 + 3187,95 = 12957,45 \text{ (руб.)}$$

Прибыль.

$$П = C_{пол} \times q_{р}, \quad (3.30)$$

где, $q_{р}$ - норматив рентабельности (15%).

$$П = 3187,95 \times 0,15 = 478,19 \text{ (руб.)}$$

Оптовая цена.

$$Ц_{опт} = C_{пол} + П, \quad (3.31)$$

$$Ц_{опт} = 3187,95 + 478,19 = 3666,14 \text{ (руб.)}$$

Налог на добавленную стоимость (НДС).

$$\text{НДС} = \text{Ц}_{\text{опт}} \times q_{\text{ндс}}, \quad (3.32)$$

где, $q_{\text{ндс}} = 18\%$ - ставка НДС.

$$\text{НДС} = 3666,14 \times 0,18 = 659,9 \text{ (руб.)}$$

Цена с учетом налога на добавленную стоимость.

$$\text{Ц}_{\text{опт с ндс}} = \text{Ц}_{\text{опт}} + \text{НДС}, \quad (3.33)$$

$$\text{Ц}_{\text{опт с ндс}} = 3666,14 + 659,9 = 4326,04 \text{ (руб)}$$

Срок окупаемости.

$$T_p = \frac{Z_{\text{пр}}}{\Pi \times N_{\text{год}}}, \quad (3.34)$$

$$T_p = \frac{12957,45}{478,19 \times 30000} = 0,0009 \text{ (года)}$$

Вывод

Таким образом в ходе проведенного оценочного анализа общая сумма затрат составила 12957,45 рублей, полная себестоимость - 3666,14 руб, Стоимость производства одного изделия с учетом заложенного норматива рентабельности 15% и НДС 18% составила 4326,04 руб. Срок окупаемости данного изделия составит около одного года. Для работ подобного уровня такие затраты являются приемлемыми.

Заключение

В данной бакалаврской работе была рассмотрена система управления движением беспилотного транспортного средства в пределах города, а именно движение по автомобильной парковке.

В первую очередь были изучены существующие автомобили, движущие в автоматическом режиме. В настоящее время разрабатываются и внедряются различные системы автоматической парковки, обеспечивающие параллельную или перпендикулярную парковку автомобиля в автоматическом режиме. Парковочный автопилот имеют в активе BMW, Ford, Mercedes-Benz, Nissan, Opel, Toyota, Volkswagen. Данная система позволяет экономить время водителя и избежать значительное количество ДТП и снизить утомляемость водителя.

Произведен анализ проблем возникающих у автопилота.

Выявлены следующие проблемы:

1 Датчики. Главное замедление развития автопилотов не алгоритмы, а отвратительное понимание машиной окружающего пространства. Информации с современных датчиков категорически недостаточно и совершенствование механизма принятия решений (нейросети и так далее) мало что меняет. Сканирующие лидары (черные ведерки, что стоят на большинстве прототипов, включая Google Car) за пределами дороги для серийных автомобилей и не дают необходимой для уверенного движения полноты информации.

2 Не универсальность. Аналогично Закону Парето, в автопилотах он тоже выполняется. 80% дорожных ситуаций машины понимают, оставшиеся 20% все портят.

3 Рыночная. Для автопилота важно быть не только функциональным, но и дешевым. Автомобильную навигацию создали больше двадцати лет назад. Но при цене системы в пару-тройку тысяч долларов она до сих пор ставится

менее чем на 20% всех новых автомобилей. Самоуправляемый функционал еще дороже.

Рассмотрены условия выхода на рынок автономного транспорта и методы снижения их стоимости.

Была произведена разработка алгоритмов управления автономными транспортными средствами. Разработан вариант определения нахождения автомобиля в текущий момент времени. В данной работе осуществлялся расчет коллизии, то есть проводился расчет обнаружения пересечений между собой объектов исследования. Приведен пример визуализации дороги, объекта-пилота и луча сенсоров в момент времени t . Описываются библиотеки, используемые в работе.

Проведен оценочный анализ общей суммы затрат, они составили 12957,45 рублей, полная себестоимость - 3666,14 рублей. Стоимость производства одного изделия с учетом заложенного норматива рентабельности 15% и НДС 18% составила 4326,04 руб. Срок окупаемости данного изделия составит около одного года. Для работ подобного уровня такие затраты являются приемлемыми.

Список используемых источников

- 1 Аизерман М.А. Автоматика переключения передач[Текст].- М.: Машгиз, 1948.-140 с.
- 2 Акимов Н.Н. Резисторы, конденсаторы, трансформаторы, дроссели, коммутационные устройства РЭА / Минск, 1994 г., стр. 84-97[Электронный ресурс]. Режим доступа:<http://www.minstroyrf.ru/trades/gradostroitel'naya-deyatelnost-i-arhitektura/14/>
- 3 Аксенов И.Б. Конспект лекций по дисциплине «Тепло и массообмен в РЭА»[Электронный ресурс]. Режим доступа:http://fastmb.ru/auto_shem/1169-sistema-avtonomnoy-parkovki-avtomobilya.html
- 4 Балабин И.В. Испытания автомобилей [Электронный ресурс]. Режим доступа:http://www.belsis.ru/news/popularnaja_elektronika/volvo-self-parking.html
- 5 Гришкевич А.И. Автомобили-роботы испытания, 1991.-187с [Текст].
- 6 Дрю Д. Теория транспортных потоков и управление ими. М.: Транспорт, 1972[Электронный ресурс]. Режим доступа:<http://howcarworks.ru>
- 7 Дубелир Г.Д. Городские улицы и мостовые. Киев, 1912.[Электронный ресурс]. Режим доступа:http://systemsauto.ru/active/active_park
- 8 Дульнев Г.Н. Тепло и массообмен в радиоэлектронной аппаратуре. - М.: Высш. шк., 1984, 247с.[Электронный ресурс]. Режим доступа:<https://www.autonews.ru/news/5825a0ad9a794747431209a>
- 9 Егель А.Э. Охрана труда на предприятиях автомобильного транспорта. - М.: Транспорт, 1985 -350 с. [Электронный ресурс]. Режим доступа:<https://habrahabr.ru/post/254707/>
- 10 Захарик Ю.М. Анализ способов повышения качества автоматического управления сцеплением автомобиля. Вестник машиностроения. [Электронный ресурс]. Режим доступа:http://fastmb.ru/auto_shem/1169-sistema-avtonomnoy-parkovki-avtomobilya.html

- 11 Лавренов О.П. Конструкторские расчеты при проектировании современных радиоэлектронных средств [Текст]: Учебное пособие. КГТУ им. А.Н. Туполева, 2002.
- 12 Луканин В.Н. Автотранспортные потоки и окружающая среда [Электронный ресурс]. Режим доступа: <https://geektimes.ru/post/278534/>
- 13 Луканин В. Н. Автотранспортные потоки и окружающая среда 2 [Электронный ресурс]. Режим доступа: <http://www.belsis.ru/news/popularnaja>
- 14 Новицкий Н.И. Математическая теория транспортных потоков. М.: Мир, 1966. [Электронный ресурс]. Режим доступа: <http://howcarworks.ru>
- 15 Новые ПДД РФ на 2016 год [Текст]: Правила Дорожного движения Москва : Эксмо, 2016. – 96 с.
- 16 Основы менеджмента: Организация и планирование производства. - М.: Финансы и статистика, 2008. - 208с. [Электронный ресурс]. Режим доступа: <https://www.drive2.ru/b/467869961361031198/>
- 17 Петров В.А. Автоматические системы транспортных машин - М.: Машиностроение, 1974. - 336 с. [Электронный ресурс]. Режим доступа: <http://www.novate.ru/blogs/110614/26649/>
- 18 Снакин Р.Ф. Выбор эффективных параметров контроля для системы индикации момента переключения передач, 2006 г., стр.162-174. [электронный ресурс]. Режим доступа: <https://avtojuristinfo.com/bezopasnost-avto/sistemy-aktivnoj-i-passivnoj-bezopasnosti-avtomobilya.html>
- 19 Снакин Р.Ф. Активная и пассивная безопасность автомобиля. Межвуз. сб. науч. тр. - М.: МАМИ, 2004. - С. 218 - 226. [Электронный ресурс]. Режим доступа: <http://zakon-auto.ru/info/safecar.php>
- 20 Салов А.И. Охрана труда на предприятиях автомобильного транспорта - М.: Транспорт, 1985, 350с. [Электронный ресурс]. Режим доступа: <https://ohranatruda.ru>

21Тарасик В.П. Теория движения автомобиля. – ВHV, 2006. 478с.[Электронный ресурс]. Режим доступа:<http://www.studfiles.ru/preview/2900856/>

22Трейстер Р.М. 44 источника электропитания для любительских электронных устройств[Электронный ресурс]. Режим доступа:<http://electricalschool.info/electronica/1584-vidy-jelektronnykh-ustrojjstv.html>

23Хейт Ф.П. Автотранспортные потоки и окружающая среда[Электронный ресурс]. Режим доступа:<http://www.auto-instructors.ru/school/manual/>

24Хунгуреева И.П Экономика предприятия [Текст]: Учебное пособие. - Улан-Удэ, Изд-во ВСГТУ, 2008. - 240 с.

25ЯркинаТ.В. Основы экономики предприятия[Текст]: Краткий курс. Учебное пособие для студентов вузов и средних специальных заведений. М., 2008.-170 с.

26. JohnT. Betts. Practical Methods for Optimal Control and Estimation Using NonlinearProgramming.

27. Huaiquan ZANG, Shuanyong CHEN. Electric Power Steering Simulation Analyze

28. Based on Fuzzy PID Current Tracking Control. Journal of Computational InformatioSystems, January, 2011.

28. Xin Li, Xue-Ping Zhao, Jie Chen. Controller Design for Electric Power Steering

29.System Using T-S Fuzzy Model Approach. International Journal of Automation andComputing, May 2009.

30. Qin Zhang. A generic fuzzy electrohydraulic steering controller for off-road vehicles,2003.

31. John Leonard, Jonathan How, Seth Teller. A Perception-Driven AutonomousUrban Vehicle, journal of Field Robotics, 1–48 (2008).

Приложение А

(обязательное)

Презентация бакалаврской работы



Пензенский государственный университет
Факультет машиностроения и транспорта
Кафедра «Транспортные машины»



Разработка информационной системы парковки беспилотного средства в условиях города

*Евсеева Ольга
Сергеевна
группа 13 МГ 1*

*Руководитель: к.т.н.,
доцент Генералова А.А.*

Пенза, 2017

ЦЕЛЬ ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ

Разработка бортовой информационно-измерительной и управляющей системы автоматической парковки автомобиля .

В связи с этим необходимо выполнить следующие задачи:

- 1 Рассмотреть существующие беспилотные транспортные средства;
- 2 Разработать алгоритмы траекторного управления транспортным средством, реализующих маневры с учетом возможности огибания препятствий;
- 3 Разработать программное обеспечение работы управляющей системы;
- 4 Оценить затраты на реализацию проекта.

АНАЛИЗ СУЩЕСТВУЮЩИХ АВТОНОМНЫХ ТРАНСПОРТНЫХ СРЕДСТВ

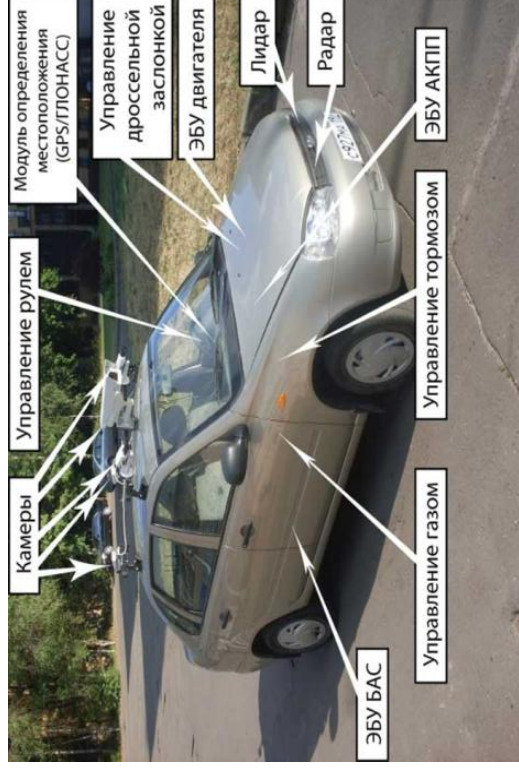
Google



АКТИВ



Лада-Калина

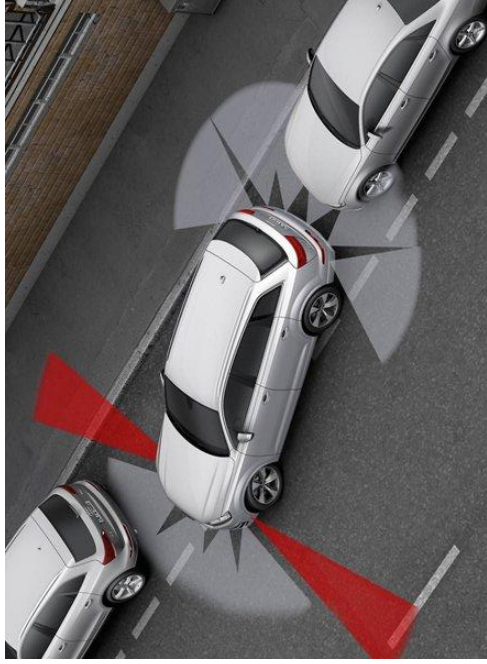


Tesla



СИСТЕМА АВТОМАТИЧЕСКОЙ И АВТОНОМНОЙ ПАРКОВКИ АВТОМОБИЛЯ

Параллельная парковка



Перпендикулярная парковка



Пульт дистанционного управления



РАЗРАБОТАННЫЕ БИБЛИОТЕКИ ФУНКЦИЙ

Пространство имен для работы с векторами

→ Ссылка: E:\CircleWalk\VectorLib.xmlcd(R)

Пространство имен для работы с элементами графики

→ Ссылка: E:\CircleWalk\DrawLib.xmlcd(R)

Пространство имен для работы с сенсорами

→ Ссылка: E:\CircleWalk\SensorLib.xmlcd(R)

Пространство имен для работы с элементами обнаружения коллизий

→ Ссылка: E:\CircleWalk\KollizLib.xmlcd(R)

Пространство имен для работы с функциями "принятия решений"

→ Ссылка: E:\CircleWalk\CalcuLib.xmlcd(R)

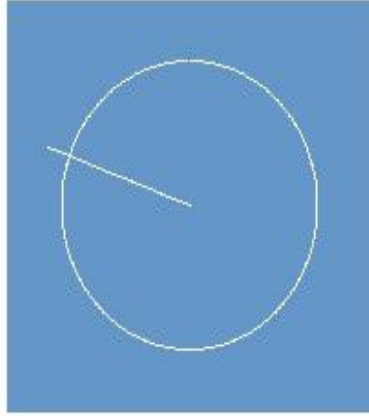
Пространство имен для работы с физикой и перемещениями

→ Ссылка: E:\CircleWalk\PhysidLib.xmlcd(R)

РАЗРАБОТКА БИБЛИОТЕКИ DROWLIB

Рисуем объект (окружность со стрелкой\линией). VectObj задан в угловых координатах

$$\begin{aligned} \text{DrowObj}(\text{PointObj}, \text{VectObj}, \text{RCircle}, \text{Img}, \text{RGB}) &:= \\ \text{Img} &\leftarrow \text{DrowSection}(\text{Img}, \text{PointObj}_0, \text{PointObj}_1, \text{RCircle}, 0, 2\pi, \text{RGB}) \\ \text{VectObj}_2 &\leftarrow \begin{pmatrix} \text{RCircle} \cdot 1.2 \\ \text{VectObj}_1 \end{pmatrix} \\ \text{Vect} &\leftarrow \text{VA2VK}(\text{VectObj}_2) \\ \text{PointCur} &\leftarrow \text{PointObj} + \text{Vect} \\ \text{Img} &\leftarrow \text{DrowLine}(\text{Img}, \text{PointObj}_0, \text{PointObj}_1, \text{PointCur}_0, \text{PointCur}_1, \text{RGB}) \\ \text{Img} \end{aligned}$$
$$\text{F2} := \text{DrowObj} \left[\begin{pmatrix} 100.5 \\ 100 \end{pmatrix}, \begin{pmatrix} 3 \\ a \cdot \frac{\pi}{180} \end{pmatrix}, 70, \text{F}, \begin{pmatrix} 255 \\ 255 \\ 255 \end{pmatrix} \right]$$



F0

РАЗРАБОТКА БИБЛИОТЕКИ VECTCLIB

Вычисляем следующее положение в соответствии с текущими показателями положения **PointObj**, вектора движения **VectObj** (в угловой форме) и вычисленного, на основе работы сенсоров, смещения **Calculate**.

time - период вычисления нового положения \ дискретность времени (уменьшить чтобы увеличить точность). По умолчанию равно 0.1 секунды

$$\text{ConditionForVector} := \begin{cases} \text{"Нижняя граница скорости"} & \text{"Верхняя граница скорости"} \\ \text{"Нижняя граница поворота"} & \text{"Верхняя граница поворота"} \end{cases}$$

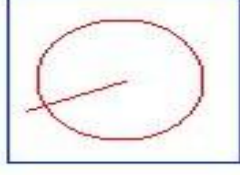
$$\begin{aligned} \text{NextPoint}(\text{PointObj}, \text{VectObj}, \text{Calculate}, \text{time}, \text{ConditionForVector}) := & \\ \text{NewVectObj} \leftarrow \text{VectObj} + \text{Calculate} & \\ \text{NewVectObj} \leftarrow \begin{cases} \text{ConditionForVector}_{0,0} \\ \text{NewVectObj}_1 \end{cases} & \text{if } \text{NewVectObj}_0 < \text{ConditionForVector}_{0,0} \\ \text{NewVectObj} \leftarrow \begin{cases} \text{ConditionForVector}_{0,1} \\ \text{NewVectObj}_1 \end{cases} & \text{if } \text{NewVectObj}_0 > \text{ConditionForVector}_{0,1} \\ \text{NewVectObj} \leftarrow \begin{cases} \text{NewVectObj}_0 \\ \text{ConditionForVector}_{1,0} + \text{VectObj}_1 \end{cases} & \text{if } \text{Calculate}_1 < \text{ConditionForVector}_{1,0} \\ \text{NewVectObj} \leftarrow \begin{cases} \text{NewVectObj}_0 \\ \text{ConditionForVector}_{1,1} + \text{VectObj}_1 \end{cases} & \text{if } \text{Calculate}_1 > \text{ConditionForVector}_{1,1} \\ \text{NewVectObj}_2 \leftarrow \text{VA2VK}(\text{NewVectObj}) & \\ \text{NewPointObj} \leftarrow \text{PointObj} + \text{NewVectObj}_2 \cdot \text{time} & \\ \begin{pmatrix} \text{NewVectObj} \\ \text{NewPointObj} \end{pmatrix} & \end{aligned}$$

РАЗРАБОТКА БИБЛИОТЕКИ KOLLIZIIV

Проверяем наличие коллизии с круглым хитбоксом в квадратной области близости.

Условие: $R_{\text{Quadrat}} > 2 * R_{\text{Circle}}$

```
KollizCircle(PointObj, FeeldKolliz, RCircle, RQuadrat) :=  
  Rq ← ceil( $\frac{R_{\text{Quadrat}}}{2}$ )  
  (a b) ← (rows(FeeldKolliz) cols(FeeldKolliz))  
  (X Y) ← (trunc(PointObj0) trunc(PointObj1))  
  Kollizia ← "false"  
  for i ∈ -Rq..Rq  
    for j ∈ -Rq..Rq  
      I ← Y + i  
      J ← X + j  
      if  $0 \leq I < a \wedge 0 \leq J < b$   
        LObj ← if (FeeldKollizI,J ≠ 0,  $\sqrt{I^2 + J^2}$ , Rq)  
        Kollizia ← "true" if LObj ≤ RCircle  
      break if Kollizia = "true"  
  Kollizia
```



РАЗРАБОТКА БИБЛИОТЕКИ SENSORLIB

Сенсоры (лучи из точки **PointSens** в направлении вектора **VectSens** (в угловой форме) на поле коллизий **FeeldKolliz**). Возвращает расстояние до объекта или конца поля **FeeldKolliz**

```
SensBeam(PointSens, VectSens, FeeldKolliz) :=  
  a ← VectSens1  
  (dx dy) ← (cos(a) sin(a))  
  (X Y) ← (PointSens0 PointSens1)  
  core ←  $\begin{pmatrix} 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 \end{pmatrix}^T$   
  Q ← "Don't Stop"  
  k ← 0  
  while Q = "Don't Stop"  
  | (X Y) ← (X + dx Y + dy)  
  | (I J) ← (trunc(Y) trunc(X))  
  | for q ∈ 0..rows(core) - 1  
  | | (i j) ← (I + coreq,0 J + coreq,1)  
  | | Q ← "Stop" if i < 0 ∨ i ≥ rows(FeeldKolliz)  
  | | Q ← "Stop" if j < 0 ∨ j ≥ cols(FeeldKolliz)  
  | | Q ← "Stop" if FeeldKollizi,j ≠ 0 if Q ≠ "Stop"  
  | k ← k + 1  
  VLong ← (X - PointSens0 Y - PointSens1)T  
  Long ← LongVect(VLong)  
  Long
```

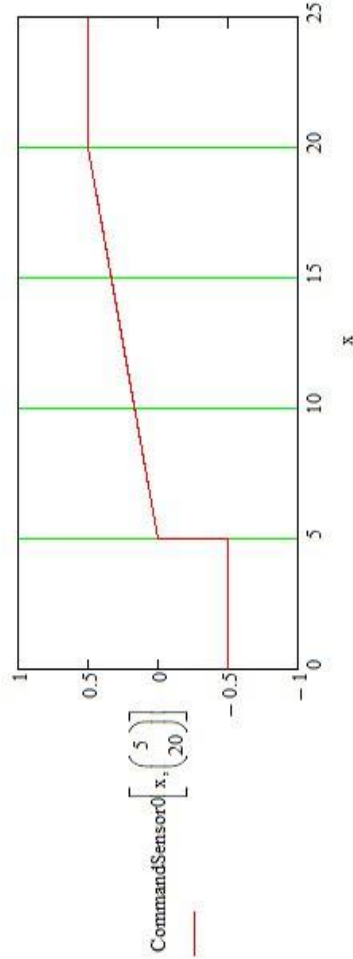
РАЗРАБОТКА БИБЛИОТЕКИ КАЛКУЛЕТИВ

$\text{ConditionForSensor0} := \begin{cases} \text{"Нижняя граница, число"} \\ \text{"Верхняя граница, число"} \end{cases}$

Выполняем следующую программу при значении сенсора SensBeam0 меньше чем нижняя граница (заданная пользователем)

Программа выполняемая при работе нулевого сенсора, ValueSensBeam0

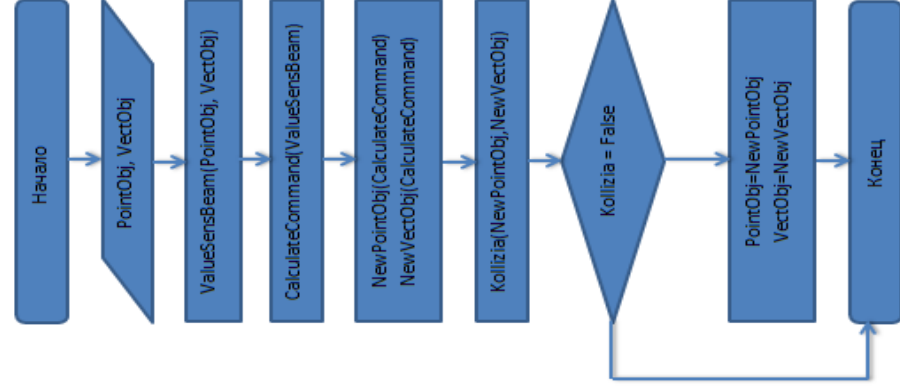
```
CommandSensor0(ValueSensBeam0, ConditionForSensor0) :=  
if ValueSensBeam0 < ConditionForSensor0_0  
| Command ← -0.5  
| Command  
if ValueSensBeam0 > ConditionForSensor0_1  
| Command ← 0.5  
| Command  
Command ← -0.5 ·  $\frac{\text{ConditionForSensor0}_1 - \text{ValueSensBeam0}}{\text{ConditionForSensor0}_1 - \text{ConditionForSensor0}_0} + 0.5$  otherwise  
| Command
```



здесь меняется скорость движения

АЛГОРИТМ РАБОТЫ ОДНОЙ ИТЕРАЦИИ ЖИЗНИ ОБЪЕКТА-ПИЛОТА

Начало
Входные данные: Положение объекта-пилота и его скорости
Получаем данные с сенсоров
Вычисляем команду на основании данных полученных с сенсоров
Вычисляем новые положение и скорость объекта-пилота исходя из команды
Обнаруживаем столкновения для новых показателей
Если столкновение есть, то переходим в конец, иначе переходим к следующему пункту
Принимаем новое положение и скорость за текущие
Конец



Live(N) :=

```

q ← 0
for t ∈ 0..N
  D ← TimeOne(RCircle, time, PointObj, VectObj, RacingWay)
  q ← q + 1 if D0 - PointObj = 0 ∧ (D1)1 = VectObj1
  if q = 5
    
$$D_1 \leftarrow D_1 + \begin{pmatrix} 0 \\ \pi \\ 60 \end{pmatrix}$$

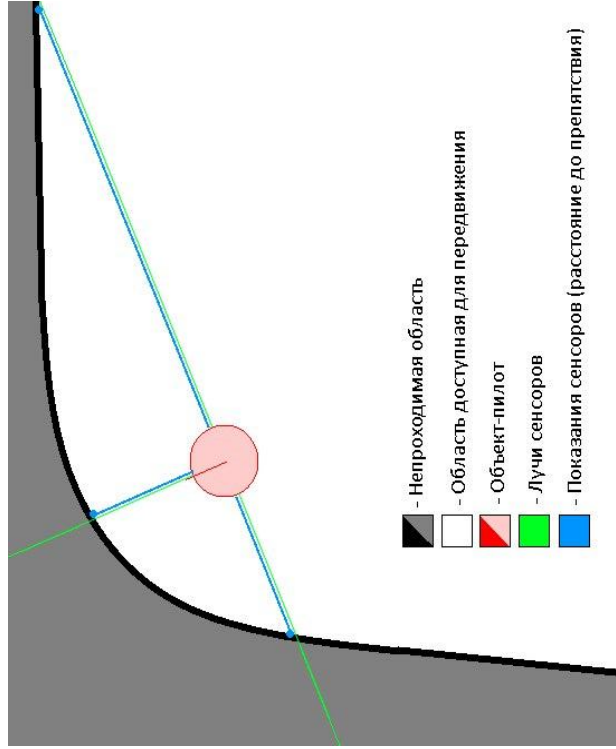
    q ← 0
  (PointObj VectObj) ← (D0 D1)
  Z ← Visual(D, RCircle)
  name ← concat("Data1\\"", num2str(t), ".jpg")
  WRITERGB(name, Z)
  break if D5 = "true"
D
  
```

Организуем цикл расчетов жизни объекта-пилота

ОБНАРУЖЕНИЕ СТОЛКНОВЕНИЙ

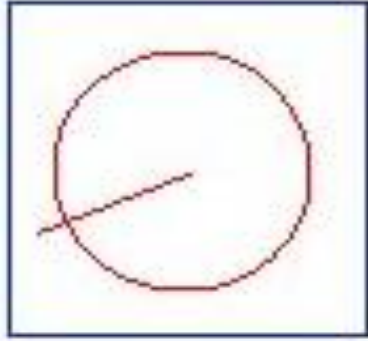
COLLISION DETECTION

Формируем пучек направляющих векторов **ListVectSens** (в угловой форме) и положений **ListPointSens** для 3-х сенсоров в зависимости от положения объекта-пилота **PointObj** и направления его взгляда\движения **VectObj**(в угловой форме). Сенсоры располагаются на поверхности хитбокса радиусом **RCircle**.



$$\text{Cluster0}(\text{PointObj}, \text{VectObj}, \text{RCircle}) :=$$

$$\begin{aligned} \text{VectSens0} &\leftarrow \left(\frac{\text{RCircle} + 1}{\text{VectObj}_1} \right) \\ \text{PointSens0} &\leftarrow \text{PointObj} + \text{VA2VK}(\text{VectSens0}) \\ \text{VectSens1} &\leftarrow \text{VectSens0} + \begin{pmatrix} 0 \\ \frac{\pi}{2} \end{pmatrix} \\ \text{PointSens1} &\leftarrow \text{PointObj} + \text{VA2VK}(\text{VectSens1}) \\ \text{VectSens2} &\leftarrow \text{VectSens0} - \begin{pmatrix} 0 \\ \frac{\pi}{2} \end{pmatrix} \\ \text{PointSens2} &\leftarrow \text{PointObj} + \text{VA2VK}(\text{VectSens2}) \\ \begin{pmatrix} \text{PointSens0} & \text{VectSens0} \\ \text{PointSens1} & \text{VectSens1} \\ \text{PointSens2} & \text{VectSens2} \end{pmatrix} \end{aligned}$$



ОБНАРУЖЕНИЕ СТОЛКНОВЕНИЯ

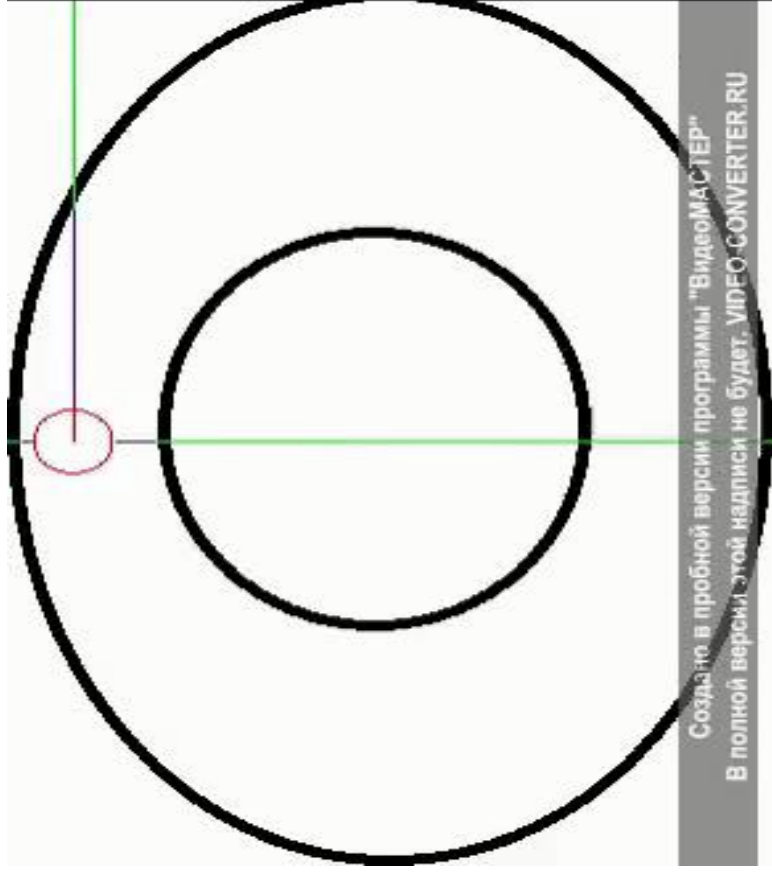
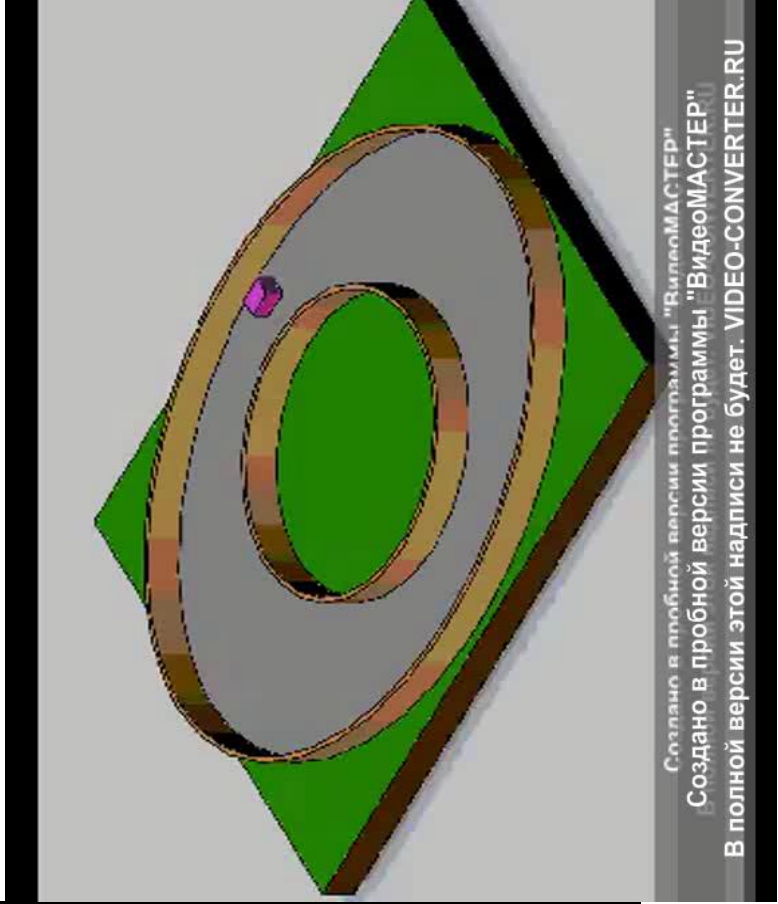


ГРАФИК ИЗМЕНЕНИЯ СКОРОСТИ В ЗАВИСИМОСТИ ОТ РАССТОЯНИЯ ДО ПРЕПЯТСТВИЯ

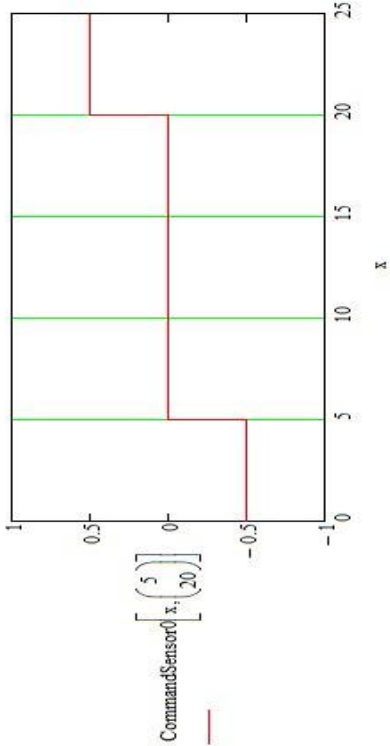
$\text{ConditionForSensor} := \left(\begin{array}{l} \text{ConditionForSensor}_0 \quad \text{ConditionForSensor}_1 \quad \text{ConditionForSensor}_2 \quad \text{ConditionForSensor}_3 \quad \text{ConditionForSensor}_4 \\ \text{ConditionForSensor}_0 \quad \text{ConditionForSensor}_1 \quad \text{ConditionForSensor}_2 \quad \text{ConditionForSensor}_3 \quad \text{ConditionForSensor}_4 \end{array} \right)$ Заполнить. Это точки деления лучей сенсоров

$\text{ListValueSensBeam} := \left(\begin{array}{l} \text{ValueSensBeam0} \\ \text{ValueSensBeam1} \\ \text{ValueSensBeam2} \\ \text{ValueSensBeam3} \\ \text{ValueSensBeam4} \end{array} \right)$ Заполнить. Это список работающих сенсоров \ участвующих в решении

$\text{ListCalculate}(\text{ListValueSensBeam}, \text{ConditionForSensor}) := \left(\begin{array}{l} \text{CommandSensor0} \left(\text{ListValueSensBeam}_0, \text{ConditionForSensor}^{(0)} \right) \quad \text{"Скорость"} \\ \text{CommandSensor1} \left(\text{ListValueSensBeam}_1, \text{ConditionForSensor}^{(1)} \right) \quad \text{"Угол"} \\ \text{CommandSensor2} \left(\text{ListValueSensBeam}_2, \text{ConditionForSensor}^{(2)} \right) \quad \text{"Угол"} \end{array} \right)$ Заполнить. Это список команд для сенсоров

Вычисляем итоговое изменение скорости и угла передвижения в зависимости от значений **ValueListCalculate** выдаваемых функцией **ListCalculate**

$\text{Calculate}(\text{ValueListCalculate}) := \left(\begin{array}{l} \text{Velocity} \leftarrow 0 \\ \text{Angle} \leftarrow 0 \\ \text{for } i \in 0 \dots \text{rows}(\text{ValueListCalculate}) - 1 \\ \quad \text{Velocity} \leftarrow \text{Velocity} + \text{ValueListCalculate}_{i,0} \text{ if } \text{ValueListCalculate}_{i,1} = \text{"Скорость"} \\ \quad \text{Angle} \leftarrow \text{Angle} + \text{ValueListCalculate}_{i,0} \text{ if } \text{ValueListCalculate}_{i,1} = \text{"Угол"} \end{array} \right) \left(\begin{array}{l} \text{Velocity} \\ \text{Angle} \end{array} \right)$



ЗАТРАТЫ НА РЕАЛИЗАЦИЮ ПРОЕКТА

Виды затрат	Расходы ,руб.
1Расчёт затрат на конструкторские разработки.	3312
2 Дополнительная заработная плата	993,6
3 Отчисления на социальные нужды	1463,9
4 Затраты на машинное время	4000
5 Накладные расходы	3125,45
6 Затраты а создание опытного образца	12957,45

ВЫВОДЫ

- 1 Произведен анализ существующих беспилотных транспортных средства;
- 2 Разработаны алгоритмы траекторного управления транспортным средством, реализующих маневры с учетом возможности огибания препятствий;
- 3 Создано программное обеспечение управляющей системы;
- 4 Просчитаны затраты на реализацию проекта.