Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«ГОРНО-АЛТАЙСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Физико-математический и инженерно-технологический институт

Кафедра математики, физики и информатики

02.03.01 Математика и компьютерные науки
Профиль «Геометрическое моделирование, топологические
методы и приложения»

Курусканова Алёна Андрияновна

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ (БАКАЛАВРСКАЯ) РАБОТА

ИСПОЛЬЗОВАНИЕ ЭЛЕМЕНТОВ КОМПЬЮТЕРНОГО ЗРЕНИЯ ДЛЯ УПРАВЛЕНИЯ МОБИЛЬНОЙ РОБОТОТЕХНИЧЕСКОЙ СИСТЕМОЙ

«Допустить к защите в ГЭК»
Директор ФМИТИ,
к.э.н., доцент
_____Е.Н. Поп
«___» _____2019г.
Заведующий кафедрой
к.ф-м.н., доцент
_____Е.А. Раенко
«_____» _____2019г.
Научный руководитель
к.т.н., доцент _____
Н.Г. Кудрявцев
Работа защищена
«____» _____ 2019г.
с оценкой _____
председатель ГЭК _____
(подпись)

Горно-Алтайск 2019

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В современном мире количество робототехнических систем постоянно увеличивается. Такие системы становятся частью не только специализированных производств и технологических процессов, но и занимают все более значимое место в нашей повседневной жизни. Одной из задач, возникающих при разработке робототехнических систем, является задача ориентации таких систем в окружающем мире. Для решения подобных задач инженеры и разработчики используют различные подходы, начиная от применения обычных сенсоров касания и простых ультразвуковых и инфракрасных дальномеров и заканчивая сложными системами технического зрения. Одному из подходов, позволяющих реализовать такую систему технического зрения, посвящена данная работа [1].

Цель данной работы заключается в разработке прототипа системы управления мобильной робототехнической платформой, использующей элементы компьютерного зрения.

Задачи данной работы:

– Исследование существующих робототехнических платформ.

– Исследование систем управления используемых, при решении подобных задач.

– Исследование доступных для реализации систем технического зрения.

– Разработка модели системы управления, использующей элементы технического зрения в качестве сенсоров обратной связи.

– Разработка структурно-функциональной модели создаваемой системы.

– Разработка программно-аппаратных средств и реализация стационарной части распределенной системы управления.

– Проведение испытаний стационарной части распределенной системы управления.

1. РОБОТОТЕХНИЧЕСКИЕ СИСТЕМЫ

В данной главе рассмотрим историю возникновения робототехники, как науки и введем определения, таких терминов, как робот, робототехнические системы и робототехника [2].

1.1 История развития робототехники

История зарождения робототехники начинается с древних времен. Начало робототехники заложил ученый Архимед, который построил аппарат «коготь», для уничтожения судов противника. Греческий механик и математик Герон Александрийский сделал такие изобретения, как измеритель святой воды, который функционировал, когда уронить в дозатор монетку, также Герон изобрел эолипил и механические двери. Эолипил представлен на Рисунке 1. Данный прибор функционировал ещё до двигателя Уатта [2].

Рисунок 1 – Паровой двигатель

Самым значимым и невероятным достижением Герона Александрийского была трёхколёсная самоходная тележка Рисунок 2, которая двигалась по определенному пути с помощью конструкции из тросов и колышков. Тележка Герона позволяла доставлять артистов на сцену, чтобы они могли выступать.

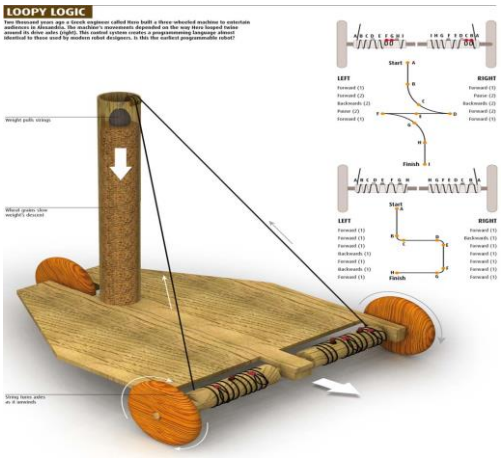

Рисунок 2 – Программируемый робот Герона

Еще одним из значимых изобретений в области робототехники является изобретенный Архитой Тарентским деревянный голубь, который показан на Рисунке 3.

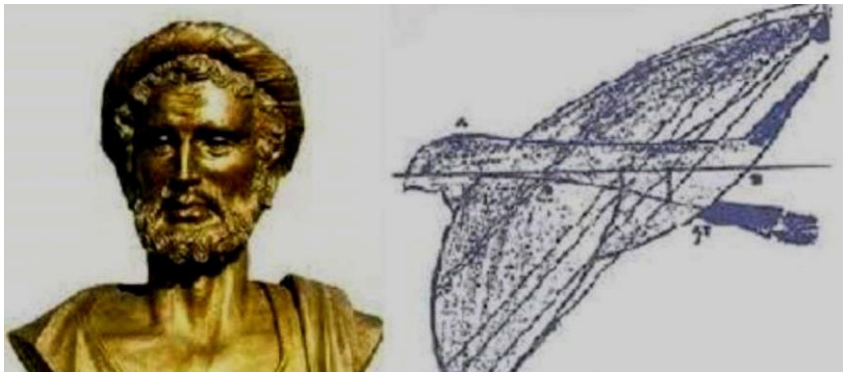

Рисунок 3 – Деревянный голубь Архита

Научно-технические разработки того времени функционировали с помощью противовесов, воды, зубчатых колес, пара, ртути, рычагов и взрывов пороха. Основными творениями ученых древности были статуи богов, которые двигали руками и шевелили головой, тем самым поражая своих зрителей.

Дальнейшим этапом становления робототехники стала эпоха Возрождения. Знаменитый богослов Альберт Великий сконструировал андроид-служанку, показанную на Рисунке 4. В тот же период были изобретены устройства, которые напоминали человека и животных. Примерами могут служить рычащие львы и певчие птицы.

Рисунок 4 – Механическая фигурка женщина-лютнистка

Робототехника развивалась не только в странах Европы, но и в других странах, например, в Персии. Развитием робототехники в Персии занимались сыновья Бану Мусы Мухаммад, Ахмад и ал-Хасан, которые сделали больше сотни различных устройств. Одной из работ братьев является «Железный мужик», который представлен на Рисунке 5.

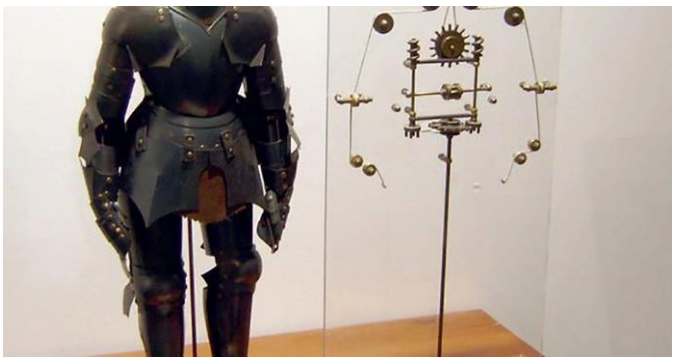

Рисунок 5 – «Железный мужик» братьев Бану Муса

Именно в истории робототехники эпоха Средневековья является самой протяженной по времени. Благодаря тому, что в те далекие времена как и в современном мире все полученные опыты, знания детально документируется, сейчас у нас есть различные описания работ и чертежей, описывающих особенности конструкций древних устройств.

Далее за эпохой Возрождения следовал Новый период, в котором было немало замечательных ученых и их работ. Примерами одних из самых

удивительных работ того времени являются творения Анри Дро и Пьера-Жака: человекоподобные фигуры, показанные на Рисунке 6, которые перемещались, записывали, изображали и воспроизводили музыку. От имени Анри Дро возникло слово «андроид» [3].

Рисунок 6 – Андроиды Анри Дро и Пьера-Жака

Далее в XIX-XX веке был разработан ткацкий станок с перфокартами. Развитие робототехники того времени происходило благодаря автоматизации промышленности. Появление электричества способствовало процветанию машиностроения. Также развивались первые роботы, так называемые андроиды, которые удивляли изобретательских людей: художников, писателей.

Автоматизация и программное управление освободили руки человека, которому представилась возможность заняться другими изобретениями. Со временем то, что являлось выдумкой, например, космический аппарат, стало привычным явлением.

Современные роботы уже могут перемещаться по воде, по воздуху, по суше, обладают «зачатками» искусственного интеллекта, ориентируются в пространстве с помощью сенсоров.

В настоящее время роботы могут принимать фактически любую форму: быть большими или маленькими, круглыми или квадратными, похожими на человека или на животное. Перспективы развития робототехники зависят от медицины, космонавтики, военного дела,

промышленности и энергетики. Роботы внедряются повсюду в сфере развлечений и, особенно в быту [3].

1.2 Что такое роботы и робототехника

Несмотря на то, что, как показывают легенды и мифы, упоминания о роботах встречались еще в древности, термин «робот» вошел в обиход только в 1920 году из пьесы Чапека Карела «Россумские Универсальные Роботы». После пьесы Чапека название «робот» стало встречаться в различных фантастических книгах, а в последующем и широко распространилось в научно-технических кругах [2].

Во второй половине 20 века появилась наука под названием робототехника. Отчасти, появление данной науки объясняется необходимостью и появившейся возможностью выполнения вместо человека, но под его контролем, тяжелых и опасных операций. Примером может служить манипулятор, работающий при высоком уровне радиации, управляемый человеком-оператором. Манипуляторы – это технические устройства, которые способны на расстоянии копировать и воспроизводить движения руки человека.

Тогда же, во второй половине 20 века, были сформулированы основные научно-технические постулаты для робототехники, автоматизации технологических процессов и компьютеризации производства [4]. Робототехника, как техническая наука, включает в себя целый ряд различных предметных областей, которые изображены на Рисунке 7.

Рисунок 7 – Предметные области робототехники

Используя термин, робототехника имеют виду практическую науку, которая исследует процессы разработки механических систем и является главной технической базой роста производства. Данная прикладная отрасль возникла на стыке кибернетики и механики.

Перечислим ниже определения двух взаимосвязанных понятий «робототехническая система» и «робот», взятых из разных источников.

Для начала дадим определения, что такое робот:

1. Робот – это машина с человекоподобными поступками, выполняющая функции животного и человека при контакте с окружающей средой.

2. Робот – это автоматическое, гидравлическое устройство, которое может реагировать конкретным действием на окружающий его мир и одновременно принимать самостоятельные действия для выполнения определенной цели.

3. Робот – автоматическая машина или несколько машин, имитирующих действия, функции человека и животных.

4. Робот – механическое и электрическое устройство, которое выполняет автономно различные алгоритмы в зависимости от меняющей вокруг окружающей обстановки [5].

Полагаясь на вышеперечисленные определения, можно сказать, что предметы, которые не чувствуют окружающий мир, роботами не являются. Это относится, например, к лампе. А, вот к роботизированным системам

относят такие устройства, которые принимают решения в зависимости от окружающей ситуации. Например, солнечная панель меняет свой угол наклона из-за того, что солнце тоже меняет свое положение.

В итоге становится ясно, что понятие «робот» разностороннее и поэтому одного определения не хватит, чтобы описать всё, что с ним связано. Поэтому, если внимательно прочитать все определения, связанные с сущностью «робот», то можно увидеть, что их объединяет одна особенность, все они моделируют действия человека при свершении работы. Благодаря, этой особенности можно различать устройства, которые не «числятся» роботами [2].

В результате вышесказанного можно сделать вывод о том, что отличие понятий робота от робототехнической системы, заключается в том, что робот представляет собой фактически одно устройство, реализованное в едином корпусе. Робототехническая система не хранится в одном месте, но управление системой производиться из одного места (с участием генерального «мозга»).

1.3 Робототехнические платформы

Как было отмечено в предыдущем разделе, робот – это электромеханическое устройство, которое может заменять людей в различных сферах деятельности, благодаря своим механическими умениям. В современном мире механические умения роботов достаточно разнообразны, поэтому человек пытается выбрать из большого перечня те функции роботов, которые предоставят ему самое выгодное решение его задачи. Чтобы найти оптимальное решение и определить наилучшее техническое устройство, необходимо сравнить действия человека и робота, произведенные при выполнении одних и тех же алгоритмов.

Роботы бывают разных типов, но мы рассмотрим самые распространённые их них:

1. Наземные роботы – колесные, гусеничные и шагающие;
2. Летающие роботы – вертолёты, дирижабль и самолеты;
3. Плавающие роботы – подводные лодки и лодки;
4. Комбинированные роботы;

Для начала рассмотрим более подробно наземный тип роботов. Наземные роботы – это самые распространённые роботы у начинающих робототехников, так как для сборки модели данного типа необходимо мало затрат.

Из наземного типа роботов самым распространённым является колесный, который можно увидеть на Рисунке 8. Данный тип объединяет роботов самых разных размеров. Диаметр колес может быть различным, например, от маленьких 4-х сантиметровых, до 40 сантиметров в диаметре. В колесном роботе может быть любое число колес, но очень часто встречается 3 и 4 колеса.

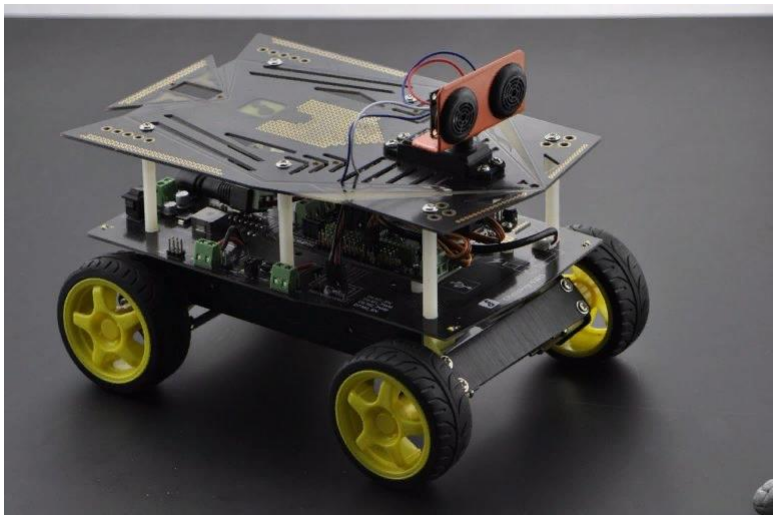

Рисунок 8 – Мобильный робот на колесах

Рассмотрим, из чего состоят трехколесные роботы. Это обычно два колеса и шаровая опора на противоположном конце. А вот, в двухколесных платформах применяется гироскопическое регулирование. Достаточно широкое применение роботов на колесах можно объяснить их существенным, по сравнению с другими конструкциями, превосходством в мобильности.

Теперь рассмотрим плюсы и минусы колесных роботов вообще:

– Преимущества – это доступная цена по сравнению с остальными типами, низкая стоимость, простой дизайн и несложное проектирование.

Далее рассмотрим, наземные роботы гусеничного типа, которые очень похожи на танк. Один из таких роботов показан на Рисунке 9. В качестве преимуществ такого типа роботов можно отметить тот факт, что вес робота распределен по всей длине конструкции. Также, за счет использования гусениц, отсутствует скольжение. Подобные «танки» могут легко преодолевать такие препятствия, как песок и гравий.

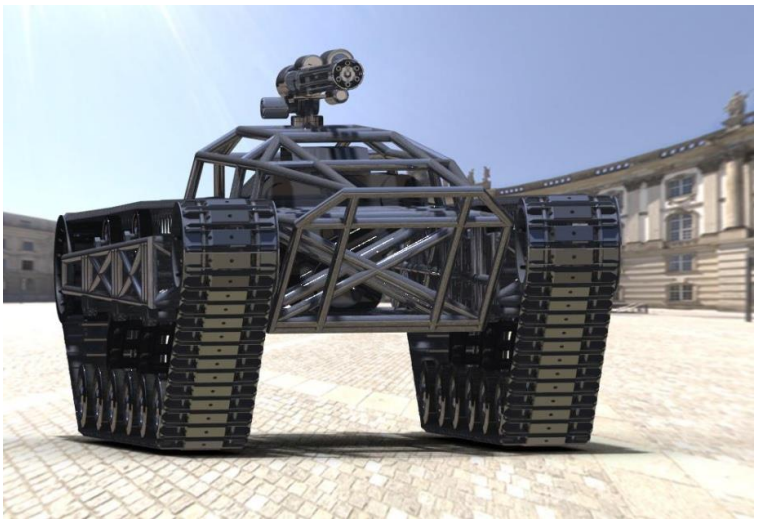

Рисунок 9 – Мобильный робот на гусеницах

Теперь выделим плюсы и минусы роботов на гусеницах:

– Преимущества: почти нет скольжения, по сравнению, с колесными роботами. Данные роботы можно применять на плоскостях с различными углами наклона.

– Недостатки: происходит достаточно быстрое изнашивание гусениц из-за бокового усилия, возникающего вследствие взаимодействия с поверхностью.

В настоящее время увеличивается число шагающих роботов, у которых для перемещения применяются ноги. Один из таких роботов изображён на Рисунке 10. Ноги у шагающих роботов используются для того, чтобы преодолевать неровности ландшафта полигона.

Рисунок 10 – Мобильный шагающий робот

Также отметим плюсы и минусы шагающих роботов:

– Преимущества: движения данных роботов похожи на естественные или натуральные. Они умеют преодолевать значительные препятствия по пересеченной местности.

– Недостатки: шагающие роботы очень дорого стоят в плане конструирования. Данные роботы имеют существенное энергопотребление и при этом требуют маленьких размеров батарей питания.

Теперь рассмотрим самый завораживающий тип роботов – это летающие роботы. В большинстве случаев под летающими роботами понимают непилотируемые летательные устройства или квадрокоптеры. Один из видов летающего робота виден на Рисунке 11.

Рисунок 11 – Мобильный летающий робот

Для военных действий используют беспилотные летальные аппараты, которые разрабатываются и изготавливаются профессионалами.

Проанализируем плюсы и минусы летающих роботов:

– Преимущества заключаются в том, что в настоящее время достаточно хорошо разработана технология изготовления и программирования летающих роботов, большинство из которых представляют собой квадрокоптеры и несложные самолеты. Существуют многочисленные группы любителей летающей техники.

– Недостатки: в результате одной аварии самолета человек может лишиться всех своих инвестиций.

Далее познакомимся с роботами, которые плавают. Речь идет о беспилотных подводных устройствах. Изучением данного типа роботов занимаются компании и институты количество, которых в настоящий момент очень быстро растет. На Рисунке 12 показан плавающий робот.

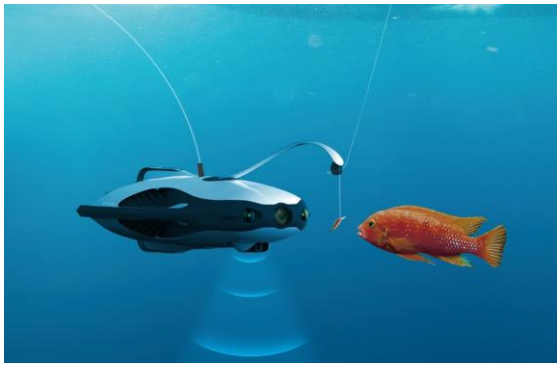

Рисунок 12 – Мобильный плавающий робот

Чтобы плавающий робот погрузился, необходимо создать балласт, также можно использовать хвост, двигатели и плавники. Радиоуправляемые лодки и катера считаются надводными моделями плавающих роботов.

Рассмотрим плюсы и минусы плавающих роботов:

– Преимущества: с использованием для данного типа роботов можно проводить многочисленные опыты и исследования, поскольку наша планета состоит из большого количества воды. Также преимуществом данного типа роботов является, то, что их испытания можно проводить в бассейне.

– Недостатки: при проведении экспериментов можно лишиться устройства в том случае, если, например, будет плохая защита электронных

составляющих от попадания воды. Также, при работе на глубинах более десяти метров, необходимы дополнительные инвестиции.

Рассмотрим последний тип роботов нашей классификации – это комбинированные мобильные роботы. К данному классу робота можно отнести в том случае, если он состоит из нескольких типов роботов. На Рисунке 13, показан комбинированный робот.

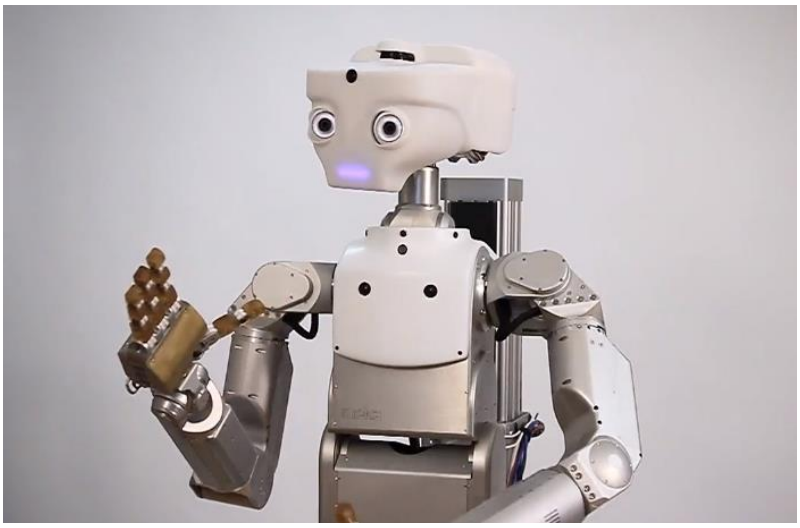

Рисунок 13 – Мобильный комбинированный робот

Проанализируем плюсы и минусы комбинированных роботов:

– Преимущества: данные роботы создаются и разрабатываются для решения конкретных целей и многочисленных задач. Они могут комплектоваться дополнительными модулями, что само по себе приводит к высокой оригинальности и функциональности конструкции.

– Недостатки: данные роботы очень сложны в сборке и поэтому у них высокая стоимость [6].

1.4 Типы управления робототехническими системами

Под управлением роботом понимается решение комплекса задач, «условно» связанных с адаптацией робота к окружающему его пространству.

Типы робототехнических систем управления (СУ) делятся на:

1. Биотехнические: командные (отдельные части робота управляются либо кнопкой, либо рычагом); копирующие (осуществляется обратная связь, робот повторяет); полуавтоматические (командный орган только управляет).

2. Автоматические: программные (роботы работают по заранее заложенной программе и окружающие их условия не меняется); адаптивные (решают стандартные задачи, но подстраиваются под условия функционирования); интеллектуальные (механические устройства, «развитые» более других устройств).

3. Интерактивные: автоматизированные (допустима смена автоматических и биотехнических режимов); супервизорные (человек в автоматических системах принимает только целеуказательные функции); диалоговые (робот советуется с человеком-оператором и принимает решения прогнозирующее поведение окружающей среды) [7].

1.4.1 Биотехнические СУ

Рассмотрим биотехнические системы управления. Это категория систем управления, в которой механическая рука робота безошибочно повторяет действия руки человека. Данный тип удобен тем, что человеку не угрожает опасность, потому что робот будет расположен далеко от оператора. Например, в случае какой-то химической аварии, когда человеку находиться смертельно опасно.

Разберём одну из подкатегорий биотехнической СУ – это командное управление. В подобных системах управление роботом в большинстве случаев происходит при помощи кнопок и рычагов (всякая кнопка выполняет определенную функцию). Преимущества данной СУ предполагает наличие обратной связи, а недостатки заключаются в том, что такая СУ подстраивается под каждого человека лично и поэтому человеку надо потратить много времени и терпения, чтобы ее реализовать.

Ещё одной подкатегорией биотехнической СУ является копирующие управление. Самым популярными представителями данного класса СУ являются экзоскелеты, которые носятся либо на всём теле, либо только на некоторых из его частей, либо только на определенной части тела. Недостатком копирующей СУ является большое потребление энергии и то, что человек в экзоскелете ограничен в движениях.

Последняя подкатегория биотехнической СУ – это полуавтоматическое управление. Для реализации данной системы управления используется микро-ЭВМ, которая вычисляет движения робота и, благодаря этому, реализуется возможность управлять кинематикой всего робота [7].

1.4.2 Автоматические СУ

Следующая категория систем управления - автоматические СУ. Данные СУ могут работать без присутствия и помощи человека. Только вначале им необходимо заложить программу поведения. Автоматических роботов используют в тех случаях, когда работа, которая выполняется роботом остается неизменной, т. е. циклически повторяется. Плюсом является, то, что не надо тратить деньги на человека, который будет программировать робота. Важный аспект использования таких роботов заключается в том, что участие человека в процессе управления снижается до минимума или вовсе отсутствует.

Обсудим одну из подкатегорий автоматической СУ – программное управление. Основная особенность данной СУ состоит в том, что робот обладает программой. Программой называется определенная последовательность операций, которая дает возможность эффективно реализовать повторение алгоритмов. Огромным плюсом данной подкатегории является, то, что программу легко и без труда можно перепрограммировать.

Далее рассмотрим еще одну подкатегорию автоматических систем управления, которая называется адаптивной. Отличие программной и адаптивной подкатегорий заключается в том, что в адаптивной подкатегории робот автономно регулирует свои операции и настраивается под меняющийся вокруг мир в соответствие с заданным алгоритмом без перепрограммирования и изменения текущей функционирующей программы.

Последней подкатегорией класса систем автоматического управления являются интеллектуальные системы управления. Заданная подкатегория – это результат усовершенствования программной и адаптивной систем управления. Отличие от других систем заключается в том, что существует обратная связь с оператором, а также связь с остальными роботами [8].

1.4.3 Интерактивные СУ

Заключительная категория – интерактивные СУ. Это комбинированные СУ, т.е. робот большую часть времени работает в автоматическом режиме, в исключительных случаях его подменяет оператор. Особенности данной категории в том, что команды роботу подаются текстом, голосом. Робот функционирует поэтапно и не переходит к другому этапу пока не получит команду-согласие от человека.

Рассмотрим подкатегории интерактивных СУ. В первой подкатегории, которую называют автоматизированной, группируются биологические и автоматические функции. В следующей супервизорской подкатегории основные отличия заключаются в том, что средства исполнения работ выбираются оператором, а на долю робота остается только роль исполнителя этих работ. В качестве последней подкатегории интерактивных СУ рассматривают диалоговые системы. В данной подкатегории СУ связь робота и оператора является неразрывной и между ними постоянно происходит диалог.

1.5 Составные части автоматической системы управления с обратной связью

На Рисунке 14 можно увидеть схематическое изображение классической версии автоматической системы управления с обратной связью.

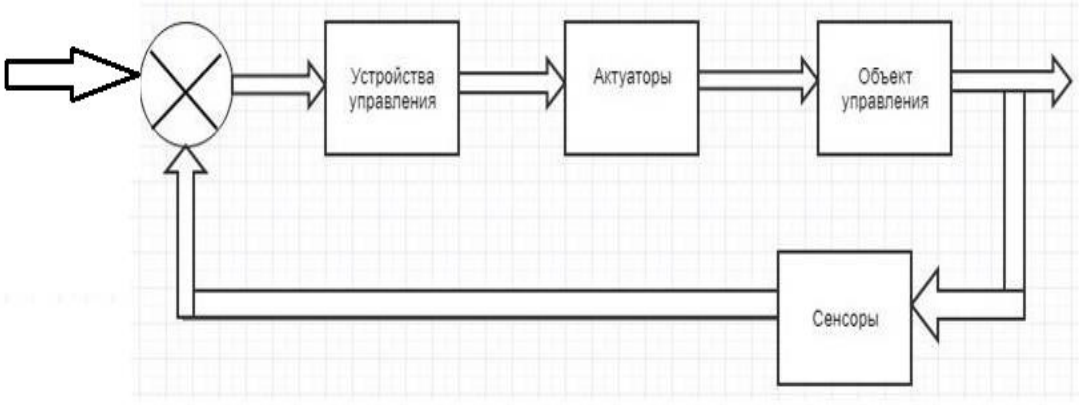

Рисунок 14 – Модель автоматической СУ с обратной связью

Рассмотрим более подробно каждый элемент данной модели. Первая составляющая – это устройство управления. Устройством управления (УУ) – в нашем случае является электронный модуль, реализованный на базе микроконтроллера, миникомпьютера или промышленного компьютера, отвечающий за реализацию алгоритма управления и функционирующий под управлением операционной системы или обычной управляющей программы [6]. На Рисунке 15 изображено устройство управления, на примере микрокомпьютера LEGO EV3 [9].

Рисунок 15 – УУ микрокомпьютера LEGO EV3

Следующим устройством в схеме системы управления является актуатор, назначение которого определяется необходимостью преобразования команд алгоритма управления в их механическое воплощение. Актуатор – это линейный механизм, который приводятся в действие электродвигателями. Электродвигатели, в свою очередь, преобразуют электрическую энергию в механическую, тем самым они выполняют полезную работу [10]. Самые популярные модели двигателей актуаторов показаны на Рисунке 16. На данном рисунке изображены коллекторные двигатели постоянного тока (1), синхронные двигатели переменного тока (2), шаговые моторы (3) и серводвигатели (4).

Рисунок 16 – Двигатели актуаторов

Следующей группой элементов автоматической системы управления с обратной связью являются сенсоры. Сенсоры предназначены для ориентации робота в окружающей среде. В качестве сенсоров (датчиков) часто используют устройства измерения, которые преобразуют одну физическую величину для лучшего измерения в другую [11]. На Рисунке 17 показаны 4 датчика: датчик влажности, датчик газа, датчик напряжения и датчик звука.

Рисунок 17 – Различные датчики Arduino

При использовании технологий компьютерного зрения в качестве сенсоров используются видеокамеры, на которые «возложена» функция «электронных глаз» робота.

1.6 Компьютерное зрение

Компьютерное зрение (машинное зрение) – это направление исследований, объединяющие теории и технологии, объединяющие роботов, которые могут отслеживать, находить и осуществлять классификацию предметов.

Как наука, машинное зрение считается теорией и технологией формирования естественных систем, которые добывают информацию с помощью изображений. Изображения могут вводиться в компьютер с помощью графических планшетов, сканеров, сенсорных экранов, WEB-камер, видеокамер, цифровых фотоаппаратов. Самой распространенной классификацией изображений являются их деление на статические и динамические [12]. Статическое изображение – это изображение, которое не меняется во время развития сцены. Динамическое изображение – это совокупность статических изображений, которые последовательно извлечены из устройства ввода [13].

Компьютерное зрение используется в различных технических системах:

– Системах управления различными процессами;
– Системах видеонаблюдения;
– Системах ввода информации;
– Системах человеко-машинного взаимодействия [14].

2. РАЗРАБОТКА И РЕАЛИЗАЦИЯ ПРОГРАММНЫХ СРЕДСТВ ДЛЯ РАСПОЗНАВАНИЯ ОБЪЕКТОВ И ОПРЕДЕЛЕНИЯ КООРДИНАТ ВЫДЕЛЕННЫХ ОБЪЕКТОВ

Одной из основных задач компьютерного зрения, является задача распознавания объекта определенного цвета или контрастного объекта (выделяющегося на общем фоне) [15].

Данная задача встречается при определении местоположения объекта или при подсчете количества объектов.

Например, роботу-футболисту (Рисунок 18) проще обнаружить оранжевый мяч на зеленом поле, чем мяч другого цвета.

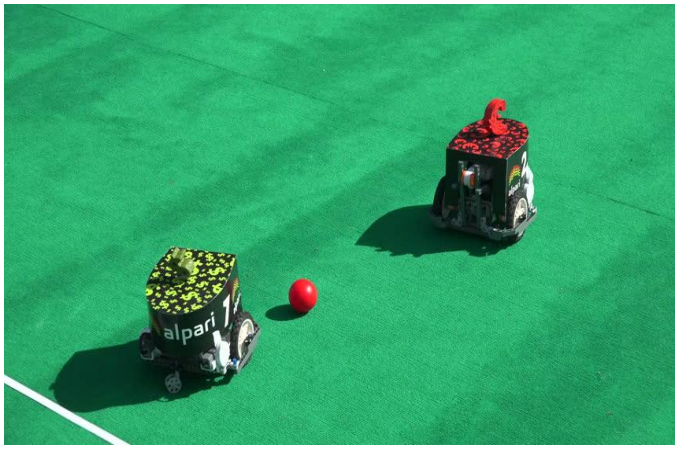

Рисунок 18 – Робот-футболист

Решение задачи обнаружения объекта по цвету в нашем случае позволит определить координаты мобильного робота в закрытом помещении или пятна лазерной засветки в электронном тире (Рисунок 19) [16].

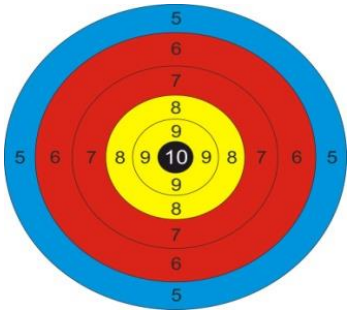

Рисунок 19 – Электронный тир

Рассмотрим возможность решения задачи обнаружения объекта по цвету с помощью специальных функций библиотеки OpenCV и языка программирования Python.

2.1 Программные средства для распознавания и определения координат контрастных объектов

В качестве языка программирования для решения задачи распознавания и определения координат контрастных объектов воспользуемся высокоуровневым языком программирования Python.

В последнее десятилетие Python приобретает все большую популярность среди программистов и применяется для разработки программного обеспечения как в небольших «настольных» проектах, так и при реализации больших программных систем [17,18].

Одной из отличительных особенностей языка программирования Python является наличие огромного количества библиотек, буквально «на все случаи жизни». Для распознавания и определения координат, используемых нами объектов было принято решение использовать одну из таких библиотек – OpenCV (Рисунок 20).

Рисунок 20 – OpenCV распознавание образов

OpenCV (Open Source Computer Vision) – это одна из библиотек, которые наиболее часто применяются для создания программного обеспечения роботов с использованием технологий машинного зрения. Функции данной библиотеки можно встраивать в программы, написанные

как на языке программирования C/C+, так и в Python программы. Библиотека OpenCV хорошо известна и поэтому сейчас уже насчитывается больше 6 миллионов скачиваний ее дистрибутива. Данная библиотека была создана русскими учеными из компании Intel [19].

2.2 Подробная установка OpenCV 2.4.12 для Python 2.7.14 в Windows

Для распознавания и определения координат контрастных объектов необходимо было установить версию OpenCV для Python. В настоящее время нет отдельного установочного комплекта библиотеки OpenCV для языка программирования Python 2.7.14. Приведем пример установки OpenCV с использованием специальной утилиты pip [20].

При выполнении данной работы была использована следующая последовательность шагов для установки Python и OpenCV (операционная система Windows):

1. Загружаем самораспаковывающийся архив с SourceForge: с сайта [21] и распаковываем его в какой-нибудь домашний каталог.

2. Устанавливаем Python 2.7.14 с сайта [22] и NumPy с сайта [23] (выбираем версию, соответствующую версии Python, мы выбрали версию `numpy-1.14.2-cp27-none-win32.whl`).

3. Копируем cv2.pyd из распакованного каталога OpenCV в `<каталог Python> \Lib\site-packages.`

4. Загружаем и устанавливаем matplotlib с сайта [24] (выбираем версию, соответствующую версии Python, мы выбрали версию `matplotlib-1.4.3-cp27-none-win32.whl`) [20].

После того, как распаковали библиотеку OpenCV, установили ее с помощью, поставляемой вместе с дистрибутивом Python утилиты pip.

Для этого в текстовом процессоре FAR мы выполнили следующую операцию:

```
pip install opencv_python-2.4.12-cp35-none-win32.whl
```

После успешного завершения данного шага проверили правильность установки. Для этого загружаем библиотечный модуль cv2 в файл проекта:

```
>>> import cv2
```

Чтобы узнать версию cv2 необходимо вести:

```
>>> cv2. __version__
'2.4.12'
```

Следующий шаг заключался в написании программы [21]. Также для корректной работы OpenCV рекомендуется установить библиотеку numpy и matplotlib:

```
pip install numpy-1.14.2-cp27-none-win32.whl
pip install matplotlib-1.4.3-cp27-none-win32.whl
```

Далее были написаны тексты программ, и приступили к их отладке.

2.3 Основные функции компьютерного зрения, связанные с видеопотоком

В самом начале рассмотрим основные функции компьютерного зрения, которые применяются при использовании видеопотока. Ниже мы покажем, как получить кадры изображения с использованием встроенной в ноутбук камеры и каким образом применить к этим кадрам преобразования: «нечеткость», «отражение» и «смена семантической модели».

Все приведенные в данной работе программы написаны на языке Python 2.7.14, а изображения получены со встроенной в ноутбук камеры [25].

Для того чтобы написать самую первую программу из приведенных нами примеров надо установить три вспомогательных Python-модуля:

```
-video.py;
-common.py;
-tst_scene_render.py.
```

Обнаружить Python-модули можно в данной папке:

```
/home/pi/Downloads/opencv-master/samples/python
```

Необходимо создать папку OpenCV обязательно в домашней папке и перекинуть туда эти Python-модули:

```
C:\Python27\Lib\site-packages\xy\z\video.py
C:\Python27\Lib\site-packages\xy\z\ common.py
C:\Python27\Lib\site-packages\xy\z\ tst_scene_render.py
```

2.3.1 Первая программа получения кадров видеопотока

Напишем первую программу, в которой в непрерывном цикле будут появляться и отображаться кадры со встроенной в ноутбук камеры. Алгоритм данной программы показан на Рисунке 21:

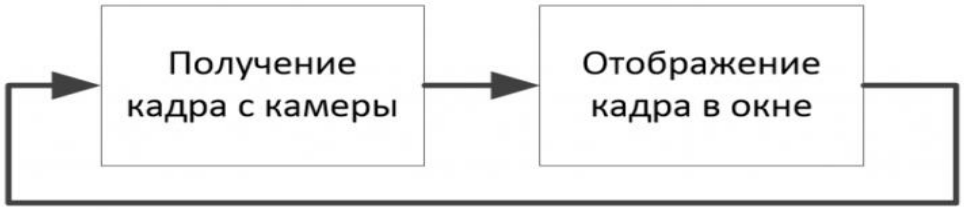

Рисунок 21 – Алгоритм программы получение кадров с камеры

Приведем пример кода программы (Приложение 1), которая выведет пробное изображение Рисунок 22, на котором изображена автор программы Курусканова Алёна Андрияновна.

Рисунок 22 – Получение кадров с камеры и вывод в окно

2.3.2 Отражение кадра в различные стороны (вверх и вниз)

В первой программе (пункта 2.3.1) были взяты кадры из видеопотока, получаемого со встроенной в ноутбук камеры и отображались в неизменном виде. Далее была добавлена еще одна операция – отражение кадра. Данную операцию необходимо выполнить сразу после получения кадра от камеры перед его отображением в окне (Рисунок 23):

Рисунок 23 – Алгоритм программы, получающей отраженные кадры

Чтобы кадр отразился необходимо использовать функцию:

flip (кадр, направление):

- *кадр* – флаг видеопотока, который мы отражаем;
- *направление* – знак, который указывает наше отражение: 0 – по-горизонтали, 1 – по-вертикали, -1 – по горизонтали и вертикали синхронно.

Пример кода программы вы можете увидеть в Приложении 2, а результат исследования виден на Рисунке 24:

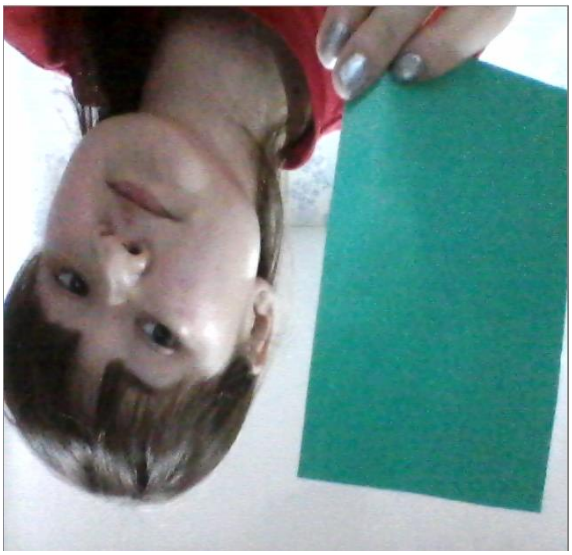

Рисунок 24 – Отражение картинки по вертикали и горизонтали

2.3.3 Изменения абстрактной модели

Теперь рассмотрим, что такое цветовая модель. Цветовая модель – это способ представления цвета кортежем из трех чисел. В качестве примера рассмотрим обозначения цветов в RGB, которые обозначаются как три составляющих: Red (красный), Green (зеленый) и Blue (синий). Также в HSV у любой составляющей существует цветовой фон – H, насыщение – S и яркость – V.

Для превращение нашей цветовой модели воспользуемся формулой:

cvtColor (кадр, модель):

- *кадр* – ссылка на текущий кадр видеопотока, который необходимо преобразовать в целевую цветовую модель;

- *модель* – целевая цветовая модель кадра, которая получится в результате преобразования.

Библиотека компьютерного зрения OpenCV содержит в себе достаточно много различных цветовых моделей, но мы при решении нашей задачи будем использовать только RGB, HSV и оттенки серого.

Пример кода программы показан в Приложении 3, а результат работы программы изображен на Рисунке 25:

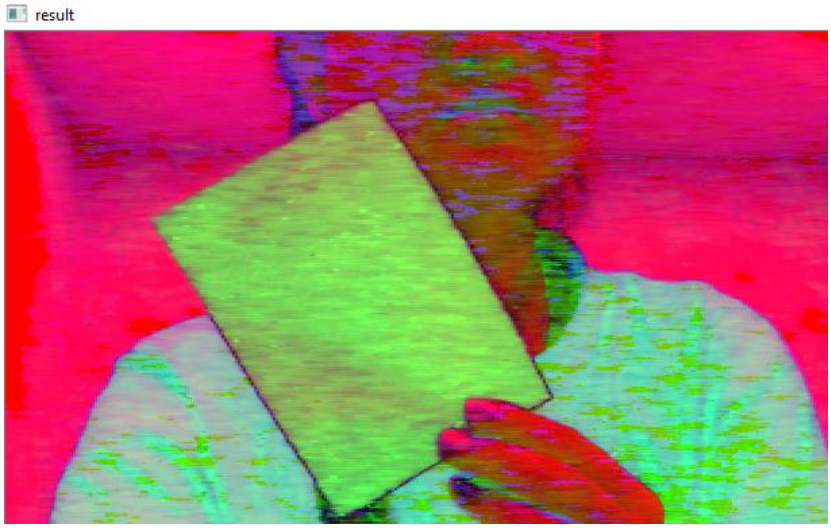

Рисунок 25 – Смена цветовой модели

2.3.4 Применение фильтра Гаусса

Иногда, когда при съемке было слабое освещение или наблюдается очень большое число шумов, то для устранения этих недостатков используют размытие по Гауссу.

Данный алгоритм состоит из трех частей:

GaussianBlur (кадр, размер ядра, отклонение):

- *кадр* – ссылка на текущий кадр видеопотока;
- *размер ядра* – размер ядра фильтра Гаусса по горизонтали и вертикали.
- *отклонение* – стандартное отклонение по оси X.

На Рисунке 26 виден результат выполнения программы, код которой размещен в Приложении 4:

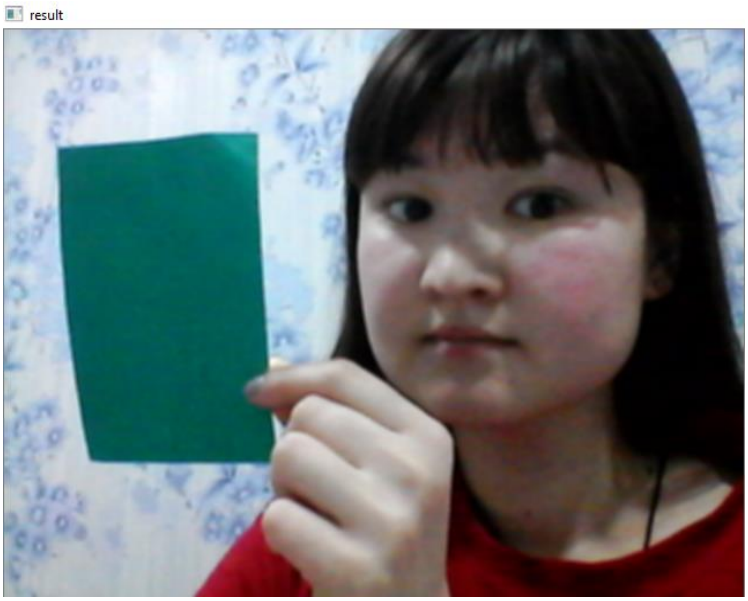

Рисунок 26 – Размытие по Гауссу

2.4 Программа для вывода текста на изображение

В предыдущем разделе были показаны основные команды, связанные с вводом изображения. Далее рассмотрим еще пару полезных функций, которые нам могут пригодится.

Первой основной функцией является вывод текста на изображение OpenCV. Чтобы получить данный наложенный текст необходимо применить следующую функцию:

putText (кадр, текст, координаты, тип шрифта, масштаб шрифта, цвет [, толщина пера [, тип линии [, центр координат]]]):

- *кадр* – изображение, на которое будет наложен текст;
- *текст* – текст, который будет показан поверх изображения;
- *координаты* – координаты левого нижнего угла текста;
- *тип шрифта* – постоянная, которую будем рассматривать внизу;
- *масштаб шрифта* –параметр, позволяющий увеличивать или уменьшать шрифт относительно стандартного;
- *цвет* – код цвета в цветовой модели RGB (цвета расположены справа налево). Рассмотрим на примере белого цвета– (255,255,255);

Приведем пример кода программы (Приложение 5), которая выведет в место кадра с координатами 20, 20 желтым цветом текст «alena kuruskanova!». Результат выполнения программы показан на Рисунке 27.

Рисунок 27 – Вывод текста поверх картинки

2.5 Программа для ввода различных геометрических фигур

Далее рассмотрим вывод различных геометрических фигур, таких как окружность и отрезок.

Первую геометрическую фигуру (отрезок) выводим на экран изображения при помощи функции Line.

line (кадр, координаты начала, координаты конца, цвет [, толщина пера [, тип линии [, сдвиг]]]):

Следующая функция (circle) нарисует нам окружность:

circle (кадр, координаты центра, радиус, цвет [, толщина пера [, тип линии [, сдвиг]]])

Прямоугольник будет нарисован при помощи функции rectangle:

rectangle (кадр, координаты 1, координаты 2, цвет [, толщина пера [, тип линии [, сдвиг]]])

Разберем программу (Приложение 6), в которой присутствуют вышеперечисленные геометрические фигуры: отрезок, окружность и прямоугольник. Результат работы программы показан на Рисунке 28.

Рисунок 28 – Вывод отрезка, окружности и прямоугольника

2.6 Реализация цветового фильтра

Одной из важных особенностей используемой нами библиотеки является наличие цветового фильтра, позволяющего распознавать различные объекты по оттенкам цветов.

Допустим, что в кадре где-то располагается зеленое яблоко (Рисунок 29).

Рисунок 29 – Зеленое яблоко

Роботу необходимо найти зеленое яблоко. Алгоритм решения данной задачи заключается в том, чтобы найти круглый объект и определить его цвет. Как можно догадаться, на изображении край яблока представляет собой окружность. Для того, чтобы точно найти яблоко, нам необходимо увидеть фрагменты изображения, на которых цвет меняется с зеленого на коричневый.

На самом деле можно упростить задачу распознавания. Для этого надо выполнить такое преобразование цвета, при котором все, что было зеленым должно стать белым, а весь остальной фон картинки должен стать черным. Благодаря этому нам не надо будет мучительно долго перебирать, и сравнивать каждый пиксель изображения, поскольку на нем будут только белые и черные точки (Рисунок 30).

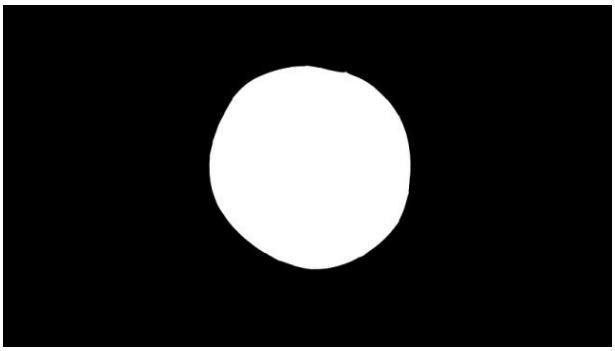

Рисунок 30 – Зеленое яблоко в черно-белом цветовом диапазоне

Вот именно такой алгоритм нам необходимо проделать с помощью OpenCV.

2.7 Цветовой диапазон в OpenCV

Чтобы наложить на кадр цветовой фильтр в определенном диапазоне необходимо использовать функцию *inRange*:

inRange (кадр, цвет 1, цвет 2):

- *кадр* – картинка, на которую закрепляем фильтр;
- *цвет 1* – первичный оттенок диапазона;
- *цвет 2* – финальный оттенок диапазона.

Для того, чтобы отметить яблоко необходимо выделить диапазон оттенков. Чтобы выделить яблоко нужен именно диапазон цветов, а не один конкретный цвет.

Пример кода программы размещен в Приложении 7, а результат исследования виден на Рисунке 31:

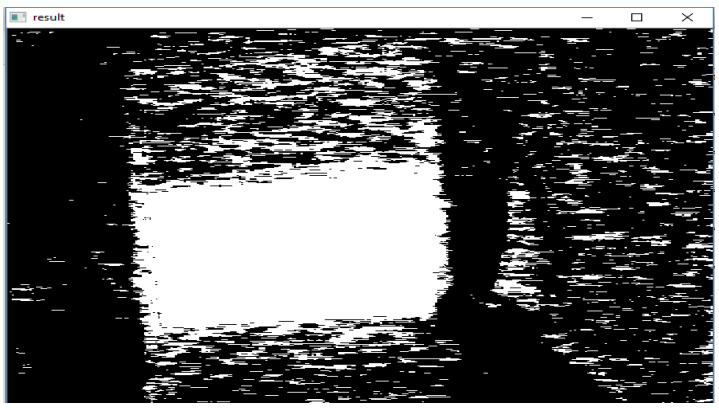

Рисунок 31 – Цветовой диапазон

2.8 Настройка цветового фильтра

Используя библиотеку OpenCV, создадим программу, которая позволит настраивать абсолютно все компоненты HSV изображения для генерирования нужного нам фильтра в режиме реального времени.

Теперь запускаем программу, код которой написан в Приложении 8, и начинаем перебирать все значения для того, чтобы вычислить первоначальный и финальный диапазон цветового фильтра. В случае фигурки в виде зеленого прямоугольника значения будут такими как показано на Рисунке 32:

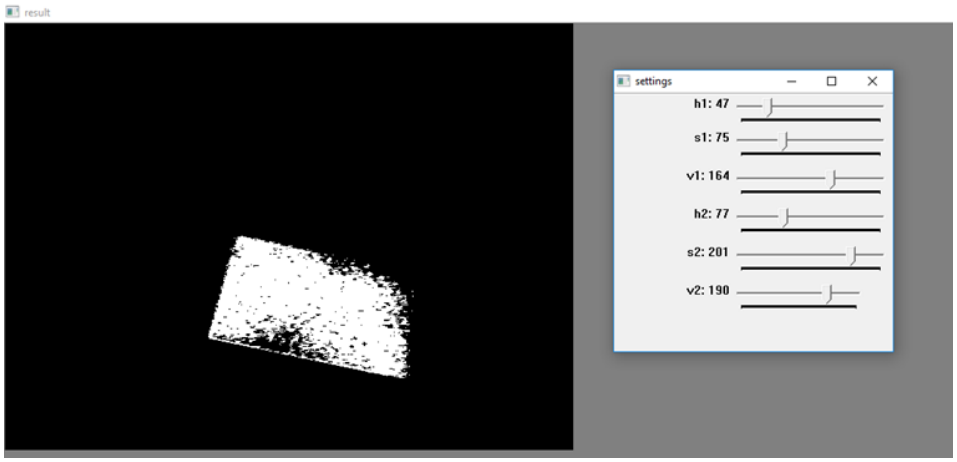

Рисунок 32 – Цветовой диапазон для зеленого прямоугольника

```
hsv_min = np.array((53, 55, 147), np.uint8)
hsv_max = np.array((83, 160, 255), np.uint8)
```

Используем полученные значения в программе из Приложения 7 и посмотрим, как фильтр определяет заданный объект.

У данного цветового фильтра существует один огромный недостаток, который заключается в том, что цвет объекта существенным образом зависит от общей освещенности: комнатной лампочки или от солнечного света.

2.9 Поиск цветового пятна с использованием OpenCV

В данном разделе постараемся изучить вопрос о том, как искать координаты цветового пятна определенного цвета на изображении. Для этого направим встроенную в ноутбук камеру на объект заданного цвета и запустим программу, которая будет определять координаты данного объекта.

Наша задача будет решаться при помощи функций библиотеки OpenCV в два этапа.

На первоначальном этапе используем цветовой фильтр и будем преобразовывать очередной кадр, полученный от камеры, в бихроматическое или бинарное изображение.

На финальном этапе воспользуемся алгоритмом вычисления моментов изображения. Для начала определим, что такое момент изображения. Моментом изображения называют интегральную характеристику пятна или, в общем случае, пятен (совокупностей пикселей, имеющих «одинаковый цвет» или одинаковую двоичную характеристику, которая указывается в качестве второго параметра в функции **moments (*кадр, двоичный*)**). Различают несколько типов моментов. Например, момент 'm00' возвратит количество всех пикселей «определенного цвета» (искомого пятна), найденных в кадре. Момент 'm01' - сумму X координат всех точек «пятна», момент 'm10' - сумму Y координат всех точек (пикселей) искомого пятна.

Рассмотрим пример, в котором найдем координаты центра искомого цветового пятна. Для того чтобы задать двоичную характеристику искомых пикселей необходимо воспользоваться специальной функцией:

```
moments (кадр, двоичный)
```

Затем необходимо запросить моменты нужных порядков. Нам необходимо три порядка это m01, m10, m00.

```
dM01 = moments ['m01']
dM10 = moments ['m10']
dArea = moments ['m00']
```

Для исключения влияния бликов перед вычислением координат середины пятна необходимо проверять условие «объемности» пятна (содержит больше 100 пикселей):

```
dArea >100
```

Если условие выполняется, то координаты X и Y центра пятна вычисляются следующим образом:

```
x = int (dM10 / dArea)
y = int (dM01 / dArea)
```

2.10 Пример программы поиска цветового пятна

Опираясь на разделы рассмотрены выше, создадим программу, которая найдет прямоугольник зеленого цвета и нарисует в нем окружность.

Приведем пример кода программы в Приложение 9, а результат исследования виден на Рисунке 33:

Рисунок 33 – Красная окружность на зеленом прямоугольнике

2.11 Вывод координат обнаруженного объекта

Еще одно усовершенствование нашей программы – это вывод координат на экран рядом с изображением найденного пятна. Для этого воспользуемся функцией PutText(), описание которой приведено в предыдущих разделах

Запускаем программу (Приложение 10) и проверяем работу алгоритма (Рисунок 34).

Рисунок 34 – Поиск цветового пятна и вывод координат

3. РЕАЛИЗАЦИЯ АВТОМАТИЧЕСКОЙ СИСТЕМЫ УПРАВЛЕНИЯ С ОБРАТНОЙ СВЯЗЬЮ С ИСПОЛЬЗОВАНИЕМ БИБЛИОТЕКИ OpenCV

В данном разделе рассмотрим конкретную реализацию системы управления, спроектированную в разделе 1.5.

3.1 Гибридная схема распределенной системы управления

Для создания устройства управления была выбрана гибридная распределенная схема. В качестве одной из частей распределенной системы управления выступал персональный компьютер (стационарная часть, ноутбук MSI Wind Notebook), а в качестве мобильной части был выбран размещенный на гусеничной робототехнической платформе микроконтроллер Arduino UNO (можно использовать Arduino Nano), который представлен на Рисунке 35.

Рисунок 35 – Микроконтроллер Arduino

В качестве актуаторов были использованы коллекторные двигатели, приводящие в движения гусеничную платформу.

В качестве объекта управления (Рисунок 36) была выбрана механическая тележка, которую можно отнести в соответствие с классификацией, представленной в главе 1 к классу гусеничных роботов.

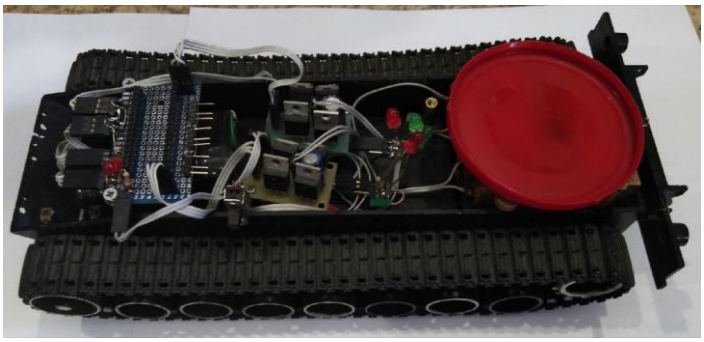

Рисунок 36 – Гусеничный робот на Arduino

Датчиком ориентации мобильной части системы управления служит электронный компас, закрепленный на платформе (Рисунок 37). Данный датчик позволяет определить стороны света, что, в свою очередь, обеспечивает корректное преобразование текущей команды, поступающей от стационарной части распределенной системы управления (выборка из таблицы coord базы данных) в команду для актуаторов гусеничной тележки.

Рисунок 37 – Электронный компас

Поскольку реализация мобильной части системы управления представляет собой стандартный проект на базе платформы Arduino, на его описании мы останавливаться не будем.

3.2 Реализация стационарной части распределенной системы управления

Далее рассмотрим структуру и реализацию стационарной части системы управления. В качестве сенсора в нашем варианте реализации стационарной части системы управления был использован установленный над полигоном на вертикальном креплении датчик ввода изображения (WEB-камера A4Tech, Рисунок 38).

Рисунок 38 – WEB-камера A4Tech

Программная реализация стационарной части сенсорной системы, отвечающей за обработку изображений, была написана на языке Python 2.7.14 библиотека OpenCV версии 2.4.12. Программный код прототипа программы размещен в Приложении 11. Условно данную программу можно разделить на два программных модуля: «Модуль обнаружения объекта и определения абсолютных координат» и «Модуль преобразования координат и формирования управляющей команды». На Рисунке 39 изображен фрагмент отладочного экрана процесса выполнения первого программного модуля, обнаруживающего объект заданного цвета (в нашем случае круг красного цвета) и определяющий его координаты.

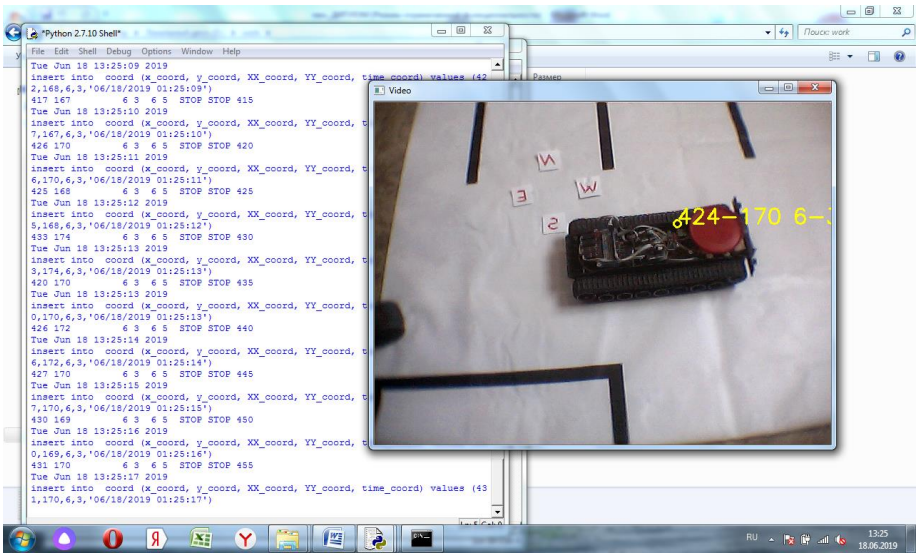

Рисунок 39 – Управляющая программа (для определения координат красных объектов)

На рисунке 40 показан фрагмент отладочного экрана процесса выполнения программного модуля, преобразующего абсолютные координаты обнаруженного объекта в их относительные образы. Также данный фрагмент программы вычисляет текущую команду, управляющую направлением движения объекта и задающую длительность перемещения.

Рисунок 40 – Фрагмент программного модуля

3.3 Реализация интерфейса взаимодействия стационарной и мобильной частей распределенной системы управления

Для реализации системы распределенного управления была решена задача разработки интерфейса взаимодействия между стационарной и мобильной частями данной системы. В качестве возможных вариантов такого интерфейса рассматривались «разделяемый» текстовый файл, сетевой сокет и таблица реляционной базы данных. После предварительного анализа было принято решение остановиться на реляционной таблице. В качестве СУБД была выбрана широко распространенная и бесплатная СУБД SQLite3. На рисунке 41 представлен стандартный интерфейс взаимодействия с SQLite3 в виде командной строки.

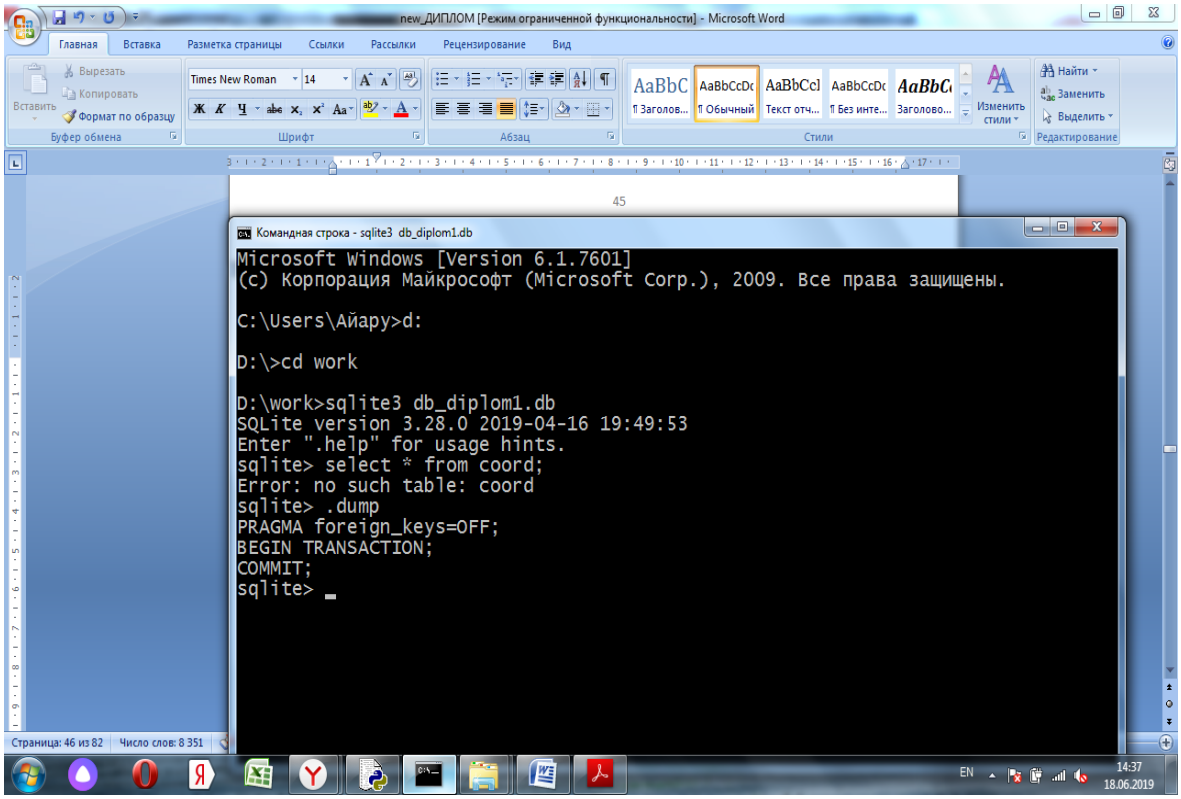

Рисунок 41 – База данных СУБД SQLite3

3.3.1 База данных (СУБД SQLite3) в качестве интерфейса взаимодействия

Ниже рассмотрим, порядок взаимодействия с СУБД SQLite3, включающий в себя создание базы данных, создание таблицы, добавление информации в таблицу и взаимодействие с таблицей, из программы, написанной на языке Python.

Для того чтобы запустить базу данных SQLite3 необходимо в командной строке написать команду «`SQLite3`» и через пробел имя базы данных, то она откроется, если нет, то будет создана в текущем каталоге.

Чтобы получить справку необходимо написать «`.help`» и в результате мы получим основные команды Таблица 1, которые нам дальнейшим пригодятся.

Таблица 1 – Основные команды СУБД SQLite3

Команда	Описание
.show	Демонстрирует реальную настройку определенных параметров
.databases	Демонстрирует название файлов и баз данных
.quit	Выход из базы данных
.tables	Демонстрирует все таблицы в данный момент
.schema	Отражает схему таблицы
.header	Заголовок базы данных
.mode	Колонки (столбцы) базы данных
.dump	Делает копию в txt

Рассмотрим структуру Таблицы 2, через которую будет передаваться информация от стационарной части системы управления к ее мобильному модулю.

Таблица 2 – Типы данных команд SQLite3

Атрибут	Тип данных	Обозначения
key_coord	INTEGER	Поле основного ключа
x_coord	INTEGER	Абсолютная координата x
y_coord	INTEGER	Абсолютная координата y
XX_coord	INTEGER	Относительная координата X
YY_coord	INTEGER	Относительная координата Y
time_coord	TEXT	Время записи
Cmd	TEXT	Команда перемещения

Для хранения координат нам необходимо создать таблицу. Назовём её coord. Выполняем команду:

```
create table coord (key_coord integer not null primary key autoincrement, x_coord integer not null, y_coord integer not null, XX_coord integer, YY_coord integer, time_coord text, cmd text).
```

Предположим, что нам необходимо, внести следующую запись Таблица 3:

Таблица 3 – Задаваемые атрибуты

x_coord	125
y_coord	230
XX_coord	65
YY_coord	115
time_coord	2019-06-18 16:18:23.234
cmd	West

Для вставки воспользуемся командой INSERT:

```
INSERT INTO coord (x_coord, y_coord, XX_coord, YY_coord, time_coord) VALUES (125, 230, 65,115, ‘2019-06-18 16:18:23.234’, ‘West’);
```

Для выборки данных воспользуемся командой SELECT:

```
select * from coord;
```

Чтобы удалить данные воспользуемся командой DELETE:

```
delete from coord;
```

Для того чтобы испытать возможность передачи данных из программы, написанной на языке Python в SQLite3 мы написали прототип программы которая представлена в Приложении 12.

3.4 Проведение экспериментов по испытанию стационарной части распределенной системы управления

После выбора всех составных частей автоматической системы управления с обратной связью и сведении базовых программных модулей, реализующих стационарную часть распределенной системы управления (модуль обнаружения объекта, модуль получения относительных координат и формирования команды, интерфейсный модуль) (Приложение 13) мы приступили к испытаниям. На вертикальную стойку установили WEB-камеру, на расстоянии требуемого для обнаружения цвета. На объекте системы управления закрепили красный круг и расположили его на полигоне. После этого запустили программу OpenCV, тем самым определив начальную точку размещения объекта в относительных координатах региона. Данная программа на OpenCV, также выдавала команды, которые были необходимы для достижения целевой точки. При проведении испытаний мы имитировали перемещения робототехнической тележки, выполняя тем самым текущие команды, сформированные системой управления.

В итоге по информации, помещенной в таблицу базы данных СУБД SQLite3 можно вычислить время перемещения робототехнической тележки между соседними регионами и общее время, требуемое для достижения цели.

Далее были проведены три эксперимента. В первом эксперименте конечную точку движения указали в квадрате региона (6-5), а начальная точка была расположена в квадрате региона (3-2) (Рисунке 42).

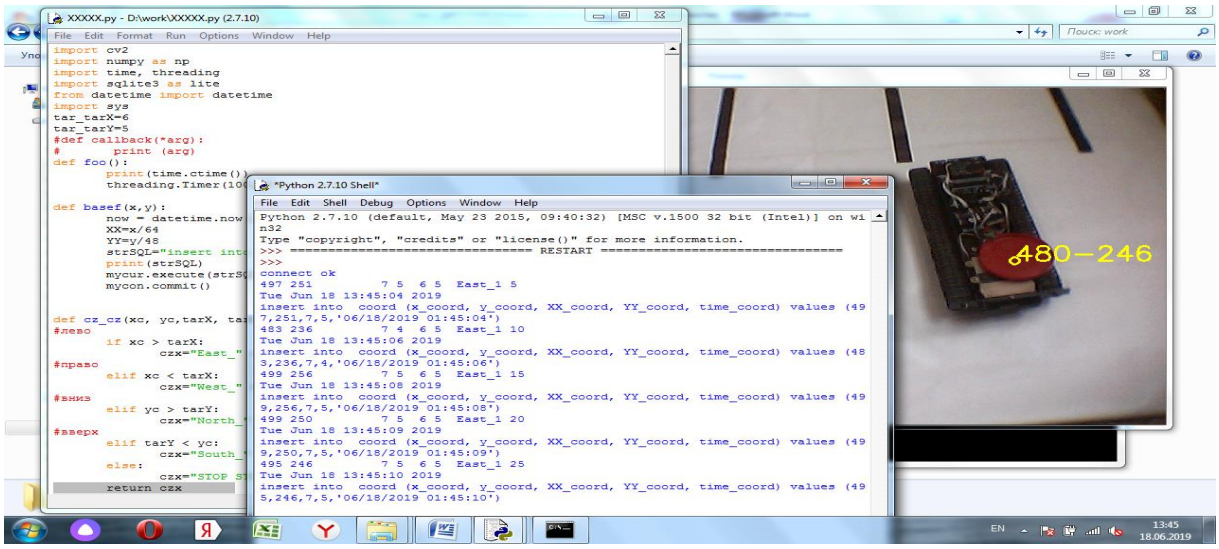

Рисунок 42 – Проведение эксперимента по испытанию стационарной части, начальная точка в квадрате региона (3-2)

Чтобы достичь квадрата региона (6-5) данной тележке потребовалось выполнить две команды West_3 и West_2. Момент достижения цели показан на Рисуноке 43.

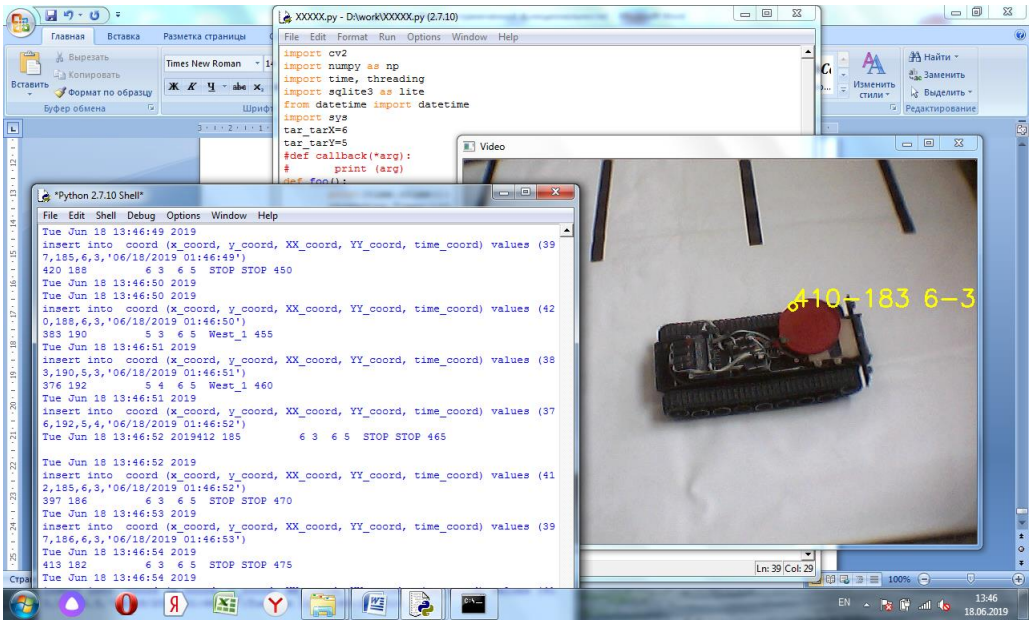

Рисунок 43 – Проведение эксперимента по испытанию стационарной части, цель в квадрате региона (6-5)

В итоге все полученные значения были записаны в базу данных СУБД SQLite3, пример листинга представлен на Рисунке 44.

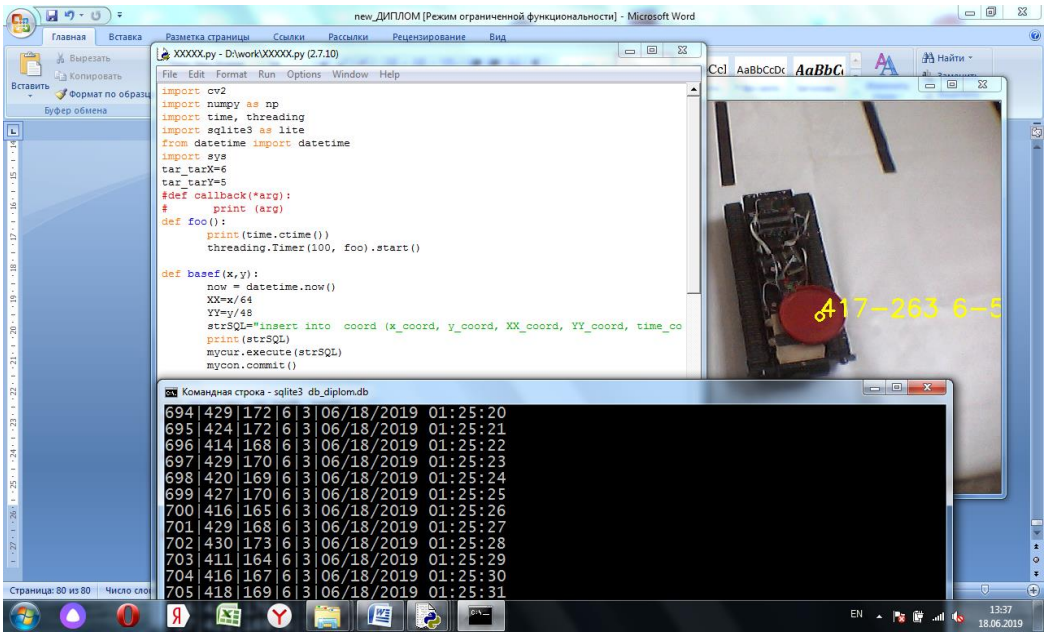

Рисунок 44 – Проведение эксперимента по испытанию стационарной части, листинг базы данных СУБД SQLite3

По информации, помещенной в таблицу базы данных СУБД SQLite3 можно увидеть скорость движения между соседними регионами, которая равна 1 секунде. Время начала первого эксперимента 01:21:47, а конец 01:25:39 (Протокол испытаний №1 – Приложение 14).

Во втором эксперименте у нас начальная точка была (4-0), а конечную, мы задали (6-5). В итоге тележка достигла конечной цели за три команды используя West_2, West_3 и East_1. Время начала эксперимента 01:36:08, а конец 01:37:53 (Протокол испытаний №2 – Приложение 15).

В последнем эксперименте мы выбрали начальную точку (4-3), цель была достигнута в результате выполнения 4 команд. Это были команды: East_1, West_4, West_2 и West_2. Время начала эксперимента 01:45:17, а конец 01:47:01 (Протокол испытаний №3 – Приложение 16).

ЗАКЛЮЧЕНИЕ

В процессе выполнения работы были исследованы и классифицированы существующие робототехнические платформы, рассмотрены системы управления, используемые при решении подобных задач, и проанализированы доступные для реализации системы технического зрения.

В результате работы создана система управления мобильной робототехнической платформой с использованием элементов компьютерного зрения. Устройство управления в данной системе было реализовано по гибридной распределенной схеме. В качестве стационарной части выступал персональный компьютер, реализация мобильной части предусматривала использование микроконтроллера на платформе Arduino. Разделяемый ресурс интерфейса взаимодействия между мобильной и стационарной частями был выполнен с использованием таблицы базы данных, реализованной на базе широко распространенной бесплатной СУБД SQLite.

Испытания показали возможность кратковременной потери объекта управления из-за недостаточно адекватной цветопередачи при разном освещении и достаточно высокую погрешность оценки местоположения объекта, связанную с низким разрешением стандартной недорогой WEB-камеры, которая была использована в проекте для ввода изображения.

Несмотря на это, полученные результаты дают основание оценивать реализуемость выбранного принципа управления, как решающего поставленную задачу и имеющего перспективу для дальнейшего развития.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ И ЛИТЕРАТУРЫ

1. Разработка масштабируемой мобильной робототехнической системы роевого взаимодействия [Электронный ресурс]: URL: https://cyberleninka.ru/article/n/razrabotka-masshtabiruemoy-mobilnoy-robototehnicheskoy -sistemy-roevogo-vzaimodeystviya (10.03.2019).

2. Робототехнические системы Торгулькин В. В., Шутков А. А. [Электронный ресурс]: URL: https://freedocs.xyz/pdf-480707185 (25.02.2019).

3. История развития робототехники [Электронный ресурс]: URL: https://legoteacher.ru/istoriya-robototehniki/istoriya-razvitiya-robototehniki.html (18.02.2019).

4. Робототехника и ее место в системе технических наук [Электронный ресурс]: URL: http://roboticslib.ru/books/item/f00/s00/z0000016/st011.shtml (18.03.2019).

5. Роботы и робототехнические устройства - термины и определения [Электронный ресурс]: URL: http://electricalschool.info/automation/2062-roboty-i-robototehnicheskie-ustroystva-terminy-i-opredeleniya.html (25.05.2019).

6. Робототехника для начинающих [Электронный ресурс]: URL: https://legoteacher.ru/10-pervyx-shagov/platforma-dlya-robota.html (23.02.2019).

7. Основы робототехники [Электронный ресурс]: URL: https://neuronus.com/theory/robo/631-osnovy-robototekhniki.html (23.02.2019).

8. Основы робототехники [Электронный ресурс]:URL: https://777russia.ru/book/uploads/АВТОМАТИКА/Юревич%20-%20Основы%20Робототехники%20-%201.pdf (29.03.2019).

9. Системы управления на микроконтроллерах [Электронный ресурс]:URL:https://studbooks.net/1883955/tovarovedenie/sistemy_upravleniya_mikrokontrollerah (29.04.2019).

10. Двигатель актуатора [Электронный ресурс]: URL: http://www.aktuator.ru/actuator_motor.shtml (20.04.2019).

11. Датчики управления перемещением роботов [Электронный ресурс]:URL: http://roboticslib.ru/books/item/f00/s00/z0000016/st033.shtml

12. Компьютерное зрение [Электронный ресурс]:URL: http://wiki-org.ru/wiki/Компьютерное_зрение (18.02.2019).

13. Обработка изображений и фильтрации статических и динамических изображений: обзор технолога [Электронный ресурс]:URL: http://masters.donntu.org/2012/fknt/shevchenko_d/library/article7.htm (18.02.2019).

14. Компьютерное зрение. Моделирование мыслительных процессов на естественном языке и Символьное моделирование [Электронный ресурс]:URL: https://intellect.ml/kompyuternoe-zrenie-1863 (18.02.2019).

15. Основные задачи компьютерного зрения [Электронный ресурс]:URL: https://lektsii.org/6-83556.html (18.02.2019).

16. Выпускная квалификационная работа магистерская диссертация [Электронный ресурс]:URL: http://elib.spbstu.ru/dl/2/v16-1652.pdf/download/v16-1652.pdf (18.03.2019).

17. Язык программирования Python [Электронный ресурс]:URL: http://inf-w.ru/?page_id=4293(05.12.2018).

18. Python [Электронный ресурс]:URL: https://ru.wikipedia.org/wiki/Python (28.03.2019).

19. OpenCV шаг за шагом. Введение. [Электронный ресурс]:URL: http://robocraft.ru/blog/computervision/264.html (28.03.2019).

20. Установка OpenCV 3.1.0 для Python 3.5 в Windows [Электронный ресурс]:URL: http://san-tit.blogspot.com/2016/03/opencv-310-python-35-windows.html (24.03.2019).

21. Open Source Computer Vision Library [Электронный ресурс]:URL: http://sourceforge.net/projects/opencvlibrary/files/opencvwin/2.4.3/ (24.04.2019).

22. Contribute to the PSF by Purchasing a PyCharm License. All proceeds benefit the PSF [Электронный ресурс]:URL: http://python.org (24.05.2019).

23. Установка scipy http://numpy.scipy.org (24.05.2019).

24. Matplotlib [Электронный ресурс]:URL: https://pypi.python.org/pypi/matplotlib/1.4.3 (24.05.2019).

25. OpenCV на python: получение кадров, смена цветовой модели и размытие [Электронный ресурс]:URL: http://robotclass.ru/tutorials/opencv-video-rgb-hsv-gray/ (24.05.2019).

26. Микроконтроллеры [Электронный ресурс]:URL: http://myowndevice.ru/index.php/theory/item/24-mikrokontrollery (24.05.2019).

27. Стартуем с SQLite3 – Основные команды [Электронный ресурс]:URL: https://ruseller.com/lessons.php?rub=28&id=2277 (10.06.2019).

ПРИЛОЖЕНИЕ 1. «Программа захвата кадров с камеры» (OpenCV, Python)

```
import cv2
import video
if __name__ == '__main__':
    # создаем окно с именем result
    cv2.namedWindow("result")
    # создаем объект cap для захвата кадров с камеры
    cap = video.create_capture(0)
    while True:
        # захватываем текущий кадр и кладем его в переменную img
        flag, img = cap.read()
        try:
            # отображаем кадр в окне с именем result
            cv2.imshow('result', img)
        except:
            cap.release()
            raise
        ch = cv2.waitKey(5)
        if ch == 27:
            break
    cap.release()
    cv2.destroyAllWindows()
```

ПРИЛОЖЕНИЕ 2. «Программа захвата и отражения кадров» (OpenCV, Python)

```
import cv2
import video
if __name__ == '__main__':
    cv2.namedWindow("result")
    cap = video.create_capture(0)
    while True:
        # захватываем текущий кадр и кладем его в переменную img
        flag, img = cap.read()
        try:
            # отражаем кадр
            img = cv2.flip(img,0)
            # отображаем кадр в окне с именем result
            cv2.imshow('result', img)
        except:
            cap.release()
            raise
        ch = cv2.waitKey(5)
        if ch == 27:
            break
    cap.release()
    cv2.destroyAllWindows()
```

ПРИЛОЖЕНИЕ 3. «Программа изменения цветовой модели кадров изображения» (OpenCV, Python)

```
import cv2
import video
if __name__ == '__main__':
    cv2.namedWindow("result")
    cap = video.create_capture(0)
    while True:
        # захватываем текущий кадр и кладем его в переменную img
        flag, img = cap.read()
        try:
            # меняем цветовую модель на HSV
            img = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
            # отображаем кадр в окне с именем result
            cv2.imshow('result', img)
        except:
            cap.release()
            raise
        ch = cv2.waitKey(5)
        if ch == 27:
            break
    cap.release()
    cv2.destroyAllWindows()
```

ПРИЛОЖЕНИЕ 4. «Программа размытия кадров изображения по Гауссу» (OpenCV, Python)

```
import cv2
import video
if __name__ == '__main__':
    # создаем окно с именем result
    cv2.namedWindow("result")
    # создаем объект cap для захвата кадров с камеры
    cap = video.create_capture(0)
    while True:
        # захватываем текущий кадр и кладем его в переменную img
        flag, img = cap.read()
        try:
            # размываем кадр
            img = cv2.GaussianBlur(img, (5, 5), 2)
            # отображаем кадр в окне с именем result
            cv2.imshow('result', img)
        except:
            cap.release()
            raise
        ch = cv2.waitKey(5)
        if ch == 27:
            break
```

ПРИЛОЖЕНИЕ 5. «Программа вывода текста на изображении» (OpenCV, Python)

```
import cv2
import video
if __name__ == '__main__':
    cv2.namedWindow("result")
cap = video.create_capture(0)
color_yellow = (0,255,255)
while True:
    flag, img = cap.read()
    try:
        cv2.putText(img, "alena kuruskanova!", (20,20),
cv2.FONT_HERSHEY_SIMPLEX, 1, color_yellow, 2)
        cv2.imshow('result', img)
    except:
        cap.release()
        raise
    ch = cv2.waitKey(5)
    if ch == 27:
        break
cap.release()
cv2.destroyAllWindows()
```

ПРИЛОЖЕНИЕ 6. «Программа рисования геометрических фигур разного цвета на изображении» (OpenCV, Python)

```
import cv2
import video
if __name__ == '__main__':
    cv2.namedWindow("result")
cap = video.create_capture(0)
color_red = (0,0,255)
color_yellow = (0,255,255)
color_purple = (255,0,255)
while True:
    flag, img = cap.read()
    try:
        # рисуем окружность
        cv2.circle(img, (190, 70), 10, color_red, -1)
        # рисуем прямоугольник
        cv2.rectangle(img, (180,140), (370,180), color_red, thickness=2, lineType=8,
shift=0)
        # рисуем пять отрезков
        for i in range (5):
            cv2.line(img, (180,85+i*5), (370,85+i*5), color_purple, thickness=2,
lineType=8, shift=0)
        # выводим текст
        cv2.putText(img, "Hello world!", (185,170),
cv2.FONT_HERSHEY_SIMPLEX, 1, color_yellow, 2)
        cv2.imshow('result', img)
    except:
        cap.release()
        raise
```

```
    ch = cv2.waitKey(5)
    if ch == 27:
        break
cap.release()
cv2.destroyAllWindows()
```

ПРИЛОЖЕНИЕ 7. «Программа получения бинарного изображения» (OpenCV, Python)

```
import cv2
import numpy as np
import video
if __name__ == '__main__':
   def callback(*arg):
       print (arg)
cv2.namedWindow("result")
cap = video.create_capture(0)
hsv_min = np.array((53, 0, 0), np.uint8)
hsv_max = np.array((83, 255, 255), np.uint8)
while True:
    flag, img = cap.read()
    hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
    thresh = cv2.inRange(hsv, hsv_min, hsv_max)
    cv2.imshow('result', thresh)
    ch = cv2.waitKey(5)
    if ch == 27:
        break
cap.release()
cv2.destroyAllWindows()
```

ПРИЛОЖЕНИЕ 8. «Программа настройки цветового фильтра» (OpenCV, Python)

```
import cv2
import numpy as np
import video
if __name__ == '__main__':
    def nothing(*arg):
        pass
cv2.namedWindow("result") # создаем главное окно
cv2.namedWindow("settings") # создаем окно настроек
cap = video.create_capture(0)
# создаем 6 бегунков для настройки начального и конечного цвета фильтра
cv2.createTrackbar('h1', 'settings', 0, 255, nothing)
cv2.createTrackbar('s1', 'settings', 0, 255, nothing)
cv2.createTrackbar('v1', 'settings', 0, 255, nothing)
cv2.createTrackbar('h2', 'settings', 255, 255, nothing)
cv2.createTrackbar('s2', 'settings', 255, 255, nothing)
cv2.createTrackbar('v2', 'settings', 255, 255, nothing)
crange = [0,0,0, 0,0,0]
while True:
    flag, img = cap.read()
    hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
    # считываем значения бегунков
    h1 = cv2.getTrackbarPos('h1', 'settings')
    s1 = cv2.getTrackbarPos('s1', 'settings')
    v1 = cv2.getTrackbarPos('v1', 'settings')
    h2 = cv2.getTrackbarPos('h2', 'settings')
    s2 = cv2.getTrackbarPos('s2', 'settings')
    v2 = cv2.getTrackbarPos('v2', 'settings')
```

```
    # формируем начальный и конечный цвет фильтра
    h_min = np.array((h1, s1, v1), np.uint8)
    h_max = np.array((h2, s2, v2), np.uint8)
    # накладываем фильтр на кадр в модели HSV
    thresh = cv2.inRange(hsv, h_min, h_max)
    cv2.imshow('result', thresh)
    ch = cv2.waitKey(5)
    if ch == 27:
        break
cap.release()
cv2.destroyAllWindows()
```

ПРИЛОЖЕНИЕ 9. «Программа обнаружения зеленого объекта» (OpenCV, Python)

```
import cv2
import numpy as np
import video
if __name__ == '__main__':
    def callback(*arg):
        print (arg)
cv2.namedWindow("result")
cap = video.create_capture(0)
# HSV фильтр для зеленых объектов из прошлого урока
hsv_min = np.array((53, 55, 147), np.uint8)
hsv_max = np.array((83, 160, 255), np.uint8)
while True:
    flag, img = cap.read()
    # преобразуем RGB картинку в HSV модель
    hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
    # применяем цветовой фильтр
    thresh = cv2.inRange(hsv, hsv_min, hsv_max)
    # вычисляем моменты изображения
    moments = cv2.moments(thresh, 1)
    dM01 = moments['m01']
    dM10 = moments['m10']
    dArea = moments['m00']
    # будем реагировать только на те моменты,
    # которые содержать больше 100 пикселей
    if dArea > 100:
        x = int (dM10 / dArea)
        y = int (dM01 / dArea)
```

```
        cv2.circle(img, (x, y), 10, (0,0,255), -1)
    cv2.imshow('result', img)
    ch = cv2.waitKey(5)
    if ch == 27:
        break
cap.release()
cv2.destroyAllWindows()
```

ПРИЛОЖЕНИЕ 10. «Программа обнаружения зеленого объекта и вывода координат его центра» (OpenCV, Python)

```
import cv2
import numpy as np
import video
if __name__ == '__main__':
    def callback(*arg):
        print (arg)
cv2.namedWindow("result")
cap = video.create_capture(0)
hsv_min = np.array((53, 55, 147), np.uint8)
hsv_max = np.array((83, 160, 255), np.uint8)
color_yellow = (0,255,255)
while True:
    flag, img = cap.read()
    img = cv2.flip(img,1) # отражение кадра вдоль оси Y
    hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
    thresh = cv2.inRange(hsv, hsv_min, hsv_max)
    moments = cv2.moments(thresh, 1)
    dM01 = moments['m01']
    dM10 = moments['m10']
    dArea = moments['m00']
    if dArea > 100:
        x = int (dM10 / dArea)
        y = int (dM01 / dArea)
        cv2.circle(img, (x, y), 5, color_yellow, 2)
        cv2.putText(img, "%d-%d" % (x,y), (x+10,y-10),
                    cv2.FONT_HERSHEY_SIMPLEX, 1, color_yellow, 2)
    cv2.imshow('result', img)
```

```
    ch = cv2.waitKey(5)
    if ch == 27:
        break
cap.release()
cv2.destroyAllWindows()
```

ПРИЛОЖЕНИЕ 11. «Программа реализации стационарной части распределенной системы управления» (OpenCV, Python)

```
import cv2
import numpy as np
import time, threading
from datetime import datetime
import sys
tar_tarX=6
tar_tarY=5
def foo():
    print(time.ctime())
    threading.Timer(100, foo).start()
def cz_cz(xc, yc,tarX, tarY):
#лево
    if xc > tarX:
        czx="East_" + str(xc-tarX)
#право
    elif xc < tarX:
        czx="West_" + str(tarX-xc)
#вниз
    elif yc > tarY:
        czx="North_" + str(yc-tarY)
#вверх
    elif tarY < yc:
        czx="South_" + str(tarY-yc)
    else:
        czx="STOP STOP"
    return czx
cap = cv2.VideoCapture(0)
```

```
# HSV фильтр для красных объектов
hsv_min = np.array((165, 90,129), np.uint8)
hsv_max = np.array((209, 195, 255), np.uint8)
color_yellow = (0,255,255)
a=0
while True:
# захватываем текущий кадр и кладем его в переменную img
    flag, img = cap. read ()
    img = cv2.flip(img,1) # отражение кадра вдоль оси Y
    hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
    thresh = cv2.inRange(hsv, hsv_min, hsv_max)
# вычисляем моменты изображения
    moments = cv2.moments(thresh, 1)
    dM01 = moments['m01']
    dM10 = moments['m10']
    dArea = moments['m00']
# будем реагировать только на те моменты,
# которые содержать больше 100 пикселей
    if dArea > 100:
        x = int(dM10 / dArea)
        y = int(dM01 / dArea)
        XXX=x/64
        YYY=y/48
        cv2.circle(img, (x, y), 5, color_yellow, 2)
        cv2.putText(img, "%d-%d %d-%d" % (x,y,XXX,YYY), (x,y),
        cv2.FONT_HERSHEY_SIMPLEX, 1, color_yellow, 2)
        a=a+1
        if a%5==0:
            cmdx=cz_cz(XXX, YYY, tar_tarX, tar_tarY)
```

```
                    ss='%d %d          %d %d %d %d %s %d' % (x, y, XXX, YYY,
tar_tarX, tar_tarY,cmdx ,a)
                    print(ss)
                    foo ()
                cv2.imshow('Video', img)
            ch = cv2.waitKey(5)
            if cv2.waitKey(1) & 0xFF == ord('q'):
                break
        mycon.close()
        cap.release()
        cv2.destroyAllWindows()
```

ПРИЛОЖЕНИЕ 12. «Программа взаимодействия с СУБД SQLite» (OpenCV, Python)

```
import cv2
import SQLite3 as lite
import sys
from datetime import datetime
def basef(x,y):
    now = datetime.now ()
    XX=x/64
    YY=y/48
    strSQL="insert into coord (x_coord, y_coord, XX_coord, YY_coord, time_coord) values (%d,%d,%d,%d,'%s')" % (x,y,XX,YY,now.strftime('%m/%d/%Y %I:%M:%S'))
    print(strSQL)
    mycur.execute(strSQL)
    mycon.commit()
mycon=lite.connect('db_diplom.db')
mycur=mycon.cursor()
print('begin')
basef(115,12)
basef(15,12)
basef(230,123)
print('Finish')
mycon. close ()
```

ПРИЛОЖЕНИЕ 13. «Программа реализации стационарной части распределенной системы управления и интерфейса взаимодействия с мобильной частью» (OpenCV, Python)

```
import cv2
import numpy as np
import time, threading
import sqlite3 as lite
from datetime import datetime
import sys
tar_tarX=6
tar_tarY=5
def foo ():
    print(time.ctime())
    threading.Timer(100, foo).start()
def basef(x,y,cmdz):
    now = datetime.now()
    XX=x/64
    YY=y/48
    strSQL="insert into coord (x_coord, y_coord, XX_coord, YY_coord, time_coord,cmd) values (%d,%d,%d,%d,'%s','%s')" % (x,y,XX,YY,now.strftime('%m/%d/%Y %I:%M:%S'),cmdz)
    print(strSQL)
    mycur.execute(strSQL)
    mycon.commit()
def cz_cz(xc, yc,tarX, tarY):
#лево
    if xc > tarX:
        czx="East_" + str(xc-tarX)
#право
```

```
        elif xc < tarX:
            czx="West_" + str(tarX-xc)
#вниз
        elif yc > tarY:
            czx="North_" + str(yc-tarY)
#вверх
        elif tarY < yc:
            czx="South_" + str(tarY-yc)
        else:
            czx="STOP STOP"
        return czx
cap = cv2.VideoCapture(0)
# HSV фильтр для красных объектов
hsv_min = np.array((165, 90,129), np.uint8)
hsv_max = np.array((209, 195, 255), np.uint8)
#КРАСНЫЙ
color_yellow = (0,255,255)
mycon=lite.connect('db_diplom.db')
mycur=mycon.cursor()
a=0
print ("connect ok")
while True:
# захватываем текущий кадр и кладем его в переменную img
    flag, img = cap.read()
    img = cv2.flip(img,1) # отражение кадра вдоль оси Y
    hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
    thresh = cv2.inRange(hsv, hsv_min, hsv_max)
# вычисляем моменты изображения
    moments = cv2.moments(thresh, 1)
    dM01 = moments['m01']
```

```
    dM10 = moments['m10']
    dArea = moments['m00']
# будем реагировать только на те моменты,
# которые содержать больше 100 пикселей
    if dArea > 100:
        x = int(dM10 / dArea)
        y = int(dM01 / dArea)
        XXX=x/64
        YYY=y/48
        cv2.circle(img, (x, y), 5, color_yellow, 2)
        cv2.putText(img, "%d-%d %d-%d" % (x,y,XXX,YYY), (x,y),
        cv2.FONT_HERSHEY_SIMPLEX, 1, color_yellow, 2)
        a=a+1
        if a%5==0:
            cmdx=cz_cz(XXX, YYY, tar_tarX, tar_tarY)
            ss='%d %d          %d %d %d %d %s %d' % (x, y, XXX, YYY, tar_tarX,
tar_tarY,cmdx ,a)
            print(ss)
            foo ()
            basef(x,y,cmdx)
        cv2.imshow('Video', img)
    ch = cv2.waitKey(5)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
mycon.close()
cap.release()
cv2.destroyAllWindows()
```

ПРИЛОЖЕНИЕ 14. «Фрагмент протокола испытания №1 стационарной части распределенной системы управления»

key_coord;x_coord;y_coord;XX_coord;YY_coord;time_coord;cmd

1369;102;281;1;5;"06/19/2019 05:27:56»; West_5

1370;102;282;1;5;"06/19/2019 05:27:58»; West_5

1371;460;406;7;8;"06/19/2019 05:27:59"; East_1

1372;236;199;3;4;"06/19/2019 05:28:00"; West_3

1373;223;182;3;3;"06/19/2019 05:28:01"; West_3

1374;169;206;2;4;"06/19/2019 05:28:02"; West_4

1375;218;183;3;3;"06/19/2019 05:28:03"; West_3

1376;168;205;2;4;"06/19/2019 05:28:09"; West_4

1377;170;203;2;4;"06/19/2019 05:28:10"; West_4

1378;168;207;2;4;"06/19/2019 05:28:11"; West_4

1379;200;191;3;3;"06/19/2019 05:28:13"; West_3

1380;167;207;2;4;"06/19/2019 05:28:14"; West_4

1381;168;206;2;4;"06/19/2019 05:28:15"; West_4

1382;236;153;3;3;"06/19/2019 05:28:16"; West_3

1383;283;199;4;4;"06/19/2019 05:28:17"; West_2

1384;397;198;6;4;"06/19/2019 05:28:18";"STOP STOP"

1385;412;197;6;4;"06/19/2019 05:28:19";"STOP STOP"

1386;412;197;6;4;"06/19/2019 05:28:21";"STOP STOP"

1387;412;197;6;4;"06/19/2019 05:28:22";"STOP STOP"

1388;410;196;6;4;"06/19/2019 05:28:23";"STOP STOP"

1389;411;196;6;4;"06/19/2019 05:28:24";"STOP STOP"

1390;412;197;6;4;"06/19/2019 05:28:25";"STOP STOP"

1391;410;196;6;4;"06/19/2019 05:28:26";"STOP STOP"

1392;411;197;6;4;"06/19/2019 05:28:27";"STOP STOP"

1393;410;196;6;4;"06/19/2019 05:28:28";"STOP STOP"

1394;409;194;6;4;"06/19/2019 05:28:30";"STOP STOP"

1395;409;195;6;4;"06/19/2019 05:28:31";"STOP STOP"
1396;409;195;6;4;"06/19/2019 05:28:32";"STOP STOP"
1397;409;196;6;4;"06/19/2019 05:28:33";"STOP STOP"
1398;409;196;6;4;"06/19/2019 05:28:34";"STOP STOP"
1399;409;195;6;4;"06/19/2019 05:28:35";"STOP STOP"
1400;410;196;6;4;"06/19/2019 05:28:36";"STOP STOP"
1401;410;195;6;4;"06/19/2019 05:28:37";"STOP STOP"

ПРИЛОЖЕНИЕ 15. «Фрагмент протокола испытания №2 стационарной части распределенной системы управления»

key_coord;x_coord;y_coord;XX_coord;YY_coord;time_coord;cmd

1402;184;105;2;2;"06/19/2019 05:30:17"; West_4

1403;187;108;2;2;"06/19/2019 05:30:18"; West_4

1404;183;105;2;2;"06/19/2019 05:30:19"; West_4

1405;186;106;2;2;"06/19/2019 05:30:20"; West_4

1406;182;106;2;2;"06/19/2019 05:30:20"; West_4

1407;190;103;2;2;"06/19/2019 05:30:21"; West_4

1408;181;105;2;2;"06/19/2019 05:30:22"; West_4

1409;193;103;3;2;"06/19/2019 05:30:22"; West_3

1410;203;101;3;2;"06/19/2019 05:30:23"; West_3

1411;195;100;3;2;"06/19/2019 05:30:24"; West_3

1412;167;107;2;2;"06/19/2019 05:30:25"; West_4

1413;147;168;2;3;"06/19/2019 05:30:26"; West_4

1414;147;163;2;3;"06/19/2019 05:30:27"; West_4

1415;134;169;2;3;"06/19/2019 05:30:27"; West_4

1416;259;131;4;2;"06/19/2019 05:30:28"; West_2

1417;132;155;2;3;"06/19/2019 05:30:29"; West_4

1418;133;196;2;4;"06/19/2019 05:30:30"; West_4

1419;131;206;2;4;"06/19/2019 05:30:31"; West_4

1420;140;165;2;3;"06/19/2019 05:30:32"; West_4

1421;165;177;2;3;"06/19/2019 05:30:33"; West_4

1422;137;213;2;4;"06/19/2019 05:30:34"; West_4

1423;208;244;3;5;"06/19/2019 05:30:35"; West_3

1424;330;250;5;5;"06/19/2019 05:30:35"; West_1

1425;333;251;5;5;"06/19/2019 05:30:36"; West_1

1426;318;246;4;5;"06/19/2019 05:30:37"; West_2

1427;336;250;5;5;"06/19/2019 05:30:38"; West_1

1428;329;253;5;5;"06/19/2019 05:30:39"; West_1
1429;333;254;5;5;"06/19/2019 05:30:39"; West_1
1430;343;262;5;5;"06/19/2019 05:30:40"; West_1
1431;313;310;4;6;"06/19/2019 05:30:41"; West_2
1432;292;379;4;7;"06/19/2019 05:30:42"; West_2
1433;312;334;4;6;"06/19/2019 05:30:43"; West_2
1434;276;353;4;7;"06/19/2019 05:30:43"; West_2
1435;330;282;5;5;"06/19/2019 05:30:44"; West_1
1436;311;289;4;6;"06/19/2019 05:30:45"; West_2
1437;460;371;7;7;"06/19/2019 05:30:45"; East_1
1438;441;370;6;7;"06/19/2019 05:30:46"; North_2
1439;445;363;6;7;"06/19/2019 05:30:47"; North_2
1440;445;369;6;7;"06/19/2019 05:30:48"; North_2
1441;452;376;7;7;"06/19/2019 05:30:49"; East_1
1442;444;377;6;7;"06/19/2019 05:30:50"; North_2
1443;415;387;6;8;"06/19/2019 05:30:50"; North_3
1444;334;340;5;7;"06/19/2019 05:30:51"; West_1
1445;300;336;4;7;"06/19/2019 05:30:52"; West_2
1446;363;301;5;6;"06/19/2019 05:30:53"; West_1
1447;472;269;7;5;"06/19/2019 05:30:54"; East_1
1448;473;271;7;5;"06/19/2019 05:30:55"; East_1
1449;470;273;7;5;"06/19/2019 05:30:56"; East_1
1450;465;269;7;5;"06/19/2019 05:30:57"; East_1
1451;472;270;7;5;"06/19/2019 05:30:57"; East_1
1452;453;271;7;5;"06/19/2019 05:30:58"; East_1
1453;477;269;7;5;"06/19/2019 05:30:59"; East_1
1454;472;271;7;5;"06/19/2019 05:31:00"; East_1
1455;453;223;7;4;"06/19/2019 05:31:01"; East_1
1456;455;223;7;4;"06/19/2019 05:31:02"; East_1
1457;452;224;7;4;"06/19/2019 05:31:03"; East_1

1458;447;223;6;4;"06/19/2019 05:31:03";"STOP STOP"

1459;374;228;5;4;"06/19/2019 05:31:04"; West_1

1460;376;227;5;4;"06/19/2019 05:31:05"; West_1

1461;376;226;5;4;"06/19/2019 05:31:06"; West_1

1462;380;228;5;4;"06/19/2019 05:31:06"; West_1

1463;388;227;6;4;"06/19/2019 05:31:07";"STOP STOP"

1464;413;230;6;4;"06/19/2019 05:31:08";"STOP STOP"

1465;405;228;6;4;"06/19/2019 05:31:09";"STOP STOP"

1466;419;230;6;4;"06/19/2019 05:31:10";"STOP STOP"

1467;417;228;6;4;"06/19/2019 05:31:11";"STOP STOP"

1468;420;231;6;4;"06/19/2019 05:31:11";"STOP STOP"

1469;420;230;6;4;"06/19/2019 05:31:12";"STOP STOP"

1470;415;241;6;5;"06/19/2019 05:31:13";"STOP STOP"

1471;416;242;6;5;"06/19/2019 05:31:14";"STOP STOP"

1472;417;241;6;5;"06/19/2019 05:31:15";"STOP STOP"

1473;418;241;6;5;"06/19/2019 05:31:15";"STOP STOP"

1474;418;241;6;5;"06/19/2019 05:31:16";"STOP STOP"

1475;417;240;6;5;"06/19/2019 05:31:17";"STOP STOP"

1476;413;241;6;5;"06/19/2019 05:31:18";"STOP STOP"

1477;413;239;6;4;"06/19/2019 05:31:19";"STOP STOP"

ПРИЛОЖЕНИЕ 16. «Фрагмент протокола испытания №3 стационарной части распределенной системы управления»

key_coord;x_coord;y_coord;XX_coord;YY_coord;time_coord;cmd

1478;437;422;6;8;"06/19/2019 05:32:26";North_3

1479;355;206;5;4;"06/19/2019 05:32:28";West_1

1480;402;172;6;3;"06/19/2019 05:32:29";"STOP STOP"

1481;168;350;2;7;"06/19/2019 05:32:33";West_4

1482;638;478;9;9;"06/19/2019 05:32:35";East_3

1483;481;118;7;2;"06/19/2019 05:32:36";East_1

1484;638;382;9;7;"06/19/2019 05:32:38";East_3

1485;239;304;3;6;"06/19/2019 05:32:40";West_3

1486;270;253;4;5;"06/19/2019 05:32:41";West_2

1487;136;184;2;3;"06/19/2019 05:32:42";West_4

1488;186;106;2;2;"06/19/2019 05:32:43";West_4

1489;239;321;3;6;"06/19/2019 05:32:44";West_3

1490;390;290;6;6;"06/19/2019 05:32:45";North_1

1491;402;302;6;6;"06/19/2019 05:32:46";North_1

1492;400;305;6;6;"06/19/2019 05:32:47";North_1

1493;263;291;4;6;"06/19/2019 05:32:48";West_2

1494;438;296;6;6;"06/19/2019 05:32:49";North_1

1495;386;286;6;5;"06/19/2019 05:32:49";"STOP STOP"

1496;347;261;5;5;"06/19/2019 05:32:50";West_1

1497;465;254;7;5;"06/19/2019 05:32:51";East_1

1498;457;264;7;5;"06/19/2019 05:32:52";East_1

1499;448;250;7;5;"06/19/2019 05:32:53";East_1

1500;444;259;6;5;"06/19/2019 05:32:54";"STOP STOP"

1501;440;254;6;5;"06/19/2019 05:32:55";"STOP STOP"

1502;376;244;5;5;"06/19/2019 05:32:55";West_1

1503;399;227;6;4;"06/19/2019 05:32:56";"STOP STOP"

1504;474;270;7;5;"06/19/2019 05:32:57";East_1

1505;468;263;7;5;"06/19/2019 05:32:58";East_1

1506;282;206;4;4;"06/19/2019 05:32:59";West_2

1507;447;269;6;5;"06/19/2019 05:33:00";"STOP STOP"

1508;459;235;7;4;"06/19/2019 05:33:01";East_1

1509;462;233;7;4;"06/19/2019 05:33:01";East_1

1510;465;234;7;4;"06/19/2019 05:33:02";East_1

1511;462;234;7;4;"06/19/2019 05:33:03";East_1

1512;406;227;6;4;"06/19/2019 05:33:04";"STOP STOP"

1513;440;280;6;5;"06/19/2019 05:33:05";"STOP STOP"

1514;458;296;7;6;"06/19/2019 05:33:06";East_1

1515;431;293;6;6;"06/19/2019 05:33:07";North_1

1516;298;284;4;5;"06/19/2019 05:33:08";West_2

1517;406;295;6;6;"06/19/2019 05:33:08";North_1

1518;502;294;7;6;"06/19/2019 05:33:09";East_1

1519;515;290;8;6;"06/19/2019 05:33:10";East_2

1520;314;244;4;5;"06/19/2019 05:33:11";West_2

1521;340;298;5;6;"06/19/2019 05:33:12";West_1

1522;370;293;5;6;"06/19/2019 05:33:13";West_1

1523;323;293;5;6;"06/19/2019 05:33:14";West_1

1524;444;302;6;6;"06/19/2019 05:33:15";North_1

1525;505;296;7;6;"06/19/2019 05:33:16";East_1

1526;351;283;5;5;"06/19/2019 05:33:16";West_1

1527;354;283;5;5;"06/19/2019 05:33:17";West_1

1528;361;286;5;5;"06/19/2019 05:33:18";West_1

1529;390;365;6;7;"06/19/2019 05:33:19";North_2

1530;421;374;6;7;"06/19/2019 05:33:20";North_2

1531;472;363;7;7;"06/19/2019 05:33:21";East_1

1532;473;298;7;6;"06/19/2019 05:33:22";East_1

1533;474;297;7;6;"06/19/2019 05:33:23";East_1

1534;471;267;7;5;"06/19/2019 05:33:24";East_1

1535;472;264;7;5;"06/19/2019 05:33:24";East_1

1536;474;266;7;5;"06/19/2019 05:33:25";East_1

1537;414;270;6;5;"06/19/2019 05:33:26";"STOP STOP"

1538;359;280;5;5;"06/19/2019 05:33:27";West_1

1539;362;282;5;5;"06/19/2019 05:33:28";West_1

1540;366;285;5;5;"06/19/2019 05:33:29";West_1

1541;369;282;5;5;"06/19/2019 05:33:30";West_1

1542;366;282;5;5;"06/19/2019 05:33:31";West_1

1543;369;282;5;5;"06/19/2019 05:33:32";West_1

1544;366;281;5;5;"06/19/2019 05:33:33";West_1

1545;377;285;5;5;"06/19/2019 05:33:34";West_1