Министерство образования и науки Российской Федерации
федеральное государственное автономное образовательное учреждение
высшего образования
**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Школа Инженерная школа информационных технологий и робототехники
Направление подготовки 15.03.06 Мехатроника и робототехника
Отделение школы (НОЦ) Отделение автоматизации и робототехники

БАКАЛАВРСКАЯ РАБОТА

Тема работы
Разработка и реализация алгоритмов распознавания изображений по видеопотоку в задачах слежения за объектами

УДК 004.93.021.621.396.965.8

Студент

Группа	ФИО	Подпись	Дата
8Е41	Волков Артем Алексеевич		

Руководитель

Должность	ФИО	Ученая степень, звание	Подпись	Дата
ассистент ОАР	Зарницын Александр Юрьевич			

КОНСУЛЬТАНТЫ:

По разделу «Финансовый менеджмент, ресурсоэффективность и ресурсосбережение»

Должность	ФИО	Ученая степень, звание	Подпись	Дата
доцент ОСГН	Петухов Олег Николаевич	к.э.н.		

По разделу «Социальная ответственность»

Должность	ФИО	Ученая степень, звание	Подпись	Дата
ассистент ОКД	Авдеева Ирина Ивановна			

ДОПУСТИТЬ К ЗАЩИТЕ:

Руководитель ООП	ФИО	Ученая степень, звание	Подпись	Дата
доцент ОАР	Мамонова Татьяна Егоровна	к.т.н.		

Томск – 2018 г.

ПЛАНИРУЕМЫЕ РЕЗУЛЬТАТЫ ОБУЧЕНИЯ ПО ООП

Код результата	Результат обучения (выпускник должен быть готов)
	Профессиональные компетенции
Р1	Применять глубокие естественно-научные, математические знания в области анализа, синтеза и проектирования для решения научных и инженерных задач производства и эксплуатации мехатронных и робототехнических устройств и систем, в том числе их систем управления
Р2	Воспринимать, обрабатывать, анализировать и обобщать научно-техническую информацию, передовой отечественный и зарубежный опыт в области теории, проектирования, производства и эксплуатации мехатронных и робототехнических устройств и систем, принимать участие в командах по разработке и эксплуатации таких устройств и систем
Р3	Применять полученные знания для решения инженерных задач при разработке, производстве и эксплуатации современных мехатронных и робототехнических устройств и систем (в том числе интеллектуальных) с использованием технологий мирового уровня, современных инструментальных и программных средств
Р4	Определять, систематизировать и получать необходимую информацию в области проектирования, производства, исследований и эксплуатации мехатронных и робототехнических модулей, устройств и систем
Р5	Планировать и проводить аналитические, имитационные и экспериментальные исследования для целей проектирования, производства и эксплуатации мехатронных и робототехнических средств и систем с использованием передового отечественного и зарубежного опыта, уметь критически оценивать полученные теоретические и экспериментальные данные и делать выводы
	Универсальные компетенции
Р6	Интегрировать знания в области анализа, проектирования, производства и эксплуатации мехатронных и робототехнических устройств и систем со знаниями из смежных областей
Р7	Понимать используемые современные методы, алгоритмы, модели и технические решения в мехатронике и робототехнике и знать области их применения, в том числе в автоматизированных производствах.
Р8	Эффективно работать в профессиональной деятельности индивидуально и в качестве члена команды
Р9	Владеть иностранным языком на уровне, позволяющем работать в интернациональной среде с пониманием культурных, языковых и социально-экономических различий
Р10	Проявлять широкую эрудицию, в том числе знание и понимание современных общественных и политических проблем, демонстрировать понимание вопросов безопасности и охраны здоровья сотрудников, юридических аспектов, ответственности за инженерную деятельность, влияния инженерных решений на социальный контекст и окружающую среду.
Р11	Следовать кодексу профессиональной этики и ответственности и международным нормам инженерной деятельности
Р12	Понимать необходимость и уметь самостоятельно учиться и повышать квалификацию в течение всего периода профессиональной деятельности

Министерство образования и науки Российской Федерации
федеральное государственное автономное образовательное учреждение
высшего образования
**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Школа Инженерная школа информационных технологий и робототехники
Направление подготовки (специальность) 15.03.06 Мехатроника и робототехника
Отделение школы (НОЦ) Отделение автоматизации и робототехники

УТВЕРЖДАЮ:
Руководитель ООП
_____ _____ Мамонова Т.Е.
(Подпись) (Дата) (Ф.И.О.)

ЗАДАНИЕ
на выполнение выпускной квалификационной работы

В форме:

Бакалаврской работы

(бакалаврской работы, дипломного проекта/работы, магистерской диссертации)

Студенту:

Группа	**ФИО**
8Е41	Волков Артем Алексеевич

Тема работы:

Разработка и реализация алгоритмов распознавания изображений по видеопотоку в задачах слежения за объектами	
Утверждена приказом директора (дата, номер)	

Срок сдачи студентом выполненной работы:	

ТЕХНИЧЕСКОЕ ЗАДАНИЕ:

Исходные данные к работе *(наименование объекта исследования или проектирования; производительность или нагрузка; режим работы (непрерывный, периодический, циклический и т. д.); вид сырья или материал изделия; требования к продукту, изделию или процессу; особые требования к особенностям функционирования (эксплуатации) объекта или изделия в плане безопасности эксплуатации, влияния на окружающую среду, энергозатратам; экономический анализ и т. д.).*	Предметом исследования является алгоритм слежения за распознанными объектами по видеопотоку. Готовая программа должна отслеживать объекты типа человек и выдавать их положение в локальных координатах камеры.

Перечень подлежащих исследованию, проектированию и разработке вопросов *(аналитический обзор по литературным источникам с целью выяснения достижений мировой науки техники в рассматриваемой области; постановка задачи исследования, проектирования, конструирования; содержание процедуры исследования, проектирования, конструирования; обсуждение результатов выполненной работы; наименование дополнительных разделов, подлежащих разработке; заключение по работе).*	1. Провести обзор литературы 2. Спроектировать архитектуру собственного приложения. 3. Написать программу на выбранном языке программирования. 4. Провести тестирование программы.
Перечень графического материала *(с точным указанием обязательных чертежей)*	1. Блок-схемы алгоритма программы. 2. Диаграммы классов. 3. Дерево вызова процедур.

Консультанты по разделам выпускной квалификационной работы

(с указанием разделов)

Раздел	Консультант
Финансовый менеджмент, ресурсоэффективность и ресурсосбережение	Петухов Олег Николаевич, доцент ОСГН, к.э.н.
Социальная ответственность	Авдеева Ирина Ивановна, ассистент ОКД

Названия разделов, которые должны быть написаны на русском и иностранном языках:

Нет

Дата выдачи задания на выполнение выпускной квалификационной работы по линейному графику	01.03.2018

Задание выдал руководитель:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
ассистент ОАР	Зарницын Александр Юрьевич			01.03.2018

Задание принял к исполнению студент:

Группа	ФИО	Подпись	Дата
8Е41	Волков Артем Алексеевич		01.03.2018

Министерство образования и науки Российской Федерации
федеральное государственное автономное образовательное учреждение
высшего образования
**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Школа Инженерная школа информационных технологий и робототехники
Направление подготовки (специальность) 15.03.06 Мехатроника и робототехника
Уровень образования бакалавр
Отделение школы (НОЦ) Отделение автоматизации и робототехники
Период выполнения (осенний / весенний семестр 2017/2018 учебного года)

Форма представления работы:

Бакалаврская работа

(бакалаврская работа, дипломный проект/работа, магистерская диссертация)

КАЛЕНДАРНЫЙ РЕЙТИНГ-ПЛАН
выполнения выпускной квалификационной работы

Срок сдачи студентом выполненной работы:	

Дата контроля	Название раздела (модуля) / вид работы (исследования)	Максимальный балл раздела (модуля)
01.06.2018	Основная часть	60
24.05.2018	Финансовый менеджмент, ресурсоэффективность и ресурсосбережение	20
16.05.2018	Социальная ответственность	20

Составил преподаватель:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
ассистент ОАР	Зарницын Александр Юрьевич			

СОГЛАСОВАНО:

Руководитель ООП	ФИО	Ученая степень, звание	Подпись	Дата
доцент ОАР	Мамонова Татьяна Егоровна	к.т.н.		

Реферат

Выпускная квалификационная работа состоит из 117 страниц, включает в себя 44 рисунков, 23 таблиц и 9 приложений. При работе были использованы 6 источников литературы.

Ключевые слова: компьютерное зрение, алгоритм слежения, тыловые роботы, мобильная робототехника, нейронная сеть, военная робототехника.

Цель работы - разработать алгоритм автоматического сопровождения человека.

Объектом исследования является практическая часть разработки программного обеспечения для видеокамеры, что реализует алгоритм слежения за распознанными объектами по видеопотоку.

Область внедрения данной работы – это программное обеспечение для тыловых роботов в воинских частях.

В работе выполнен анализ текущих разработок в области алгоритмов автоматического сопровождения для военных роботов поддержки и разработан алгоритм слежения по видеопотоку. Написанная программа была протестирована на записанных видео, в которых были типовые ситуации для данной области внедрения. По итогам тестированы были составлены количественные показатели работы программы, а также выявлены проблемы текущей реализации и предложены идеи для их устранения.

Определения

В пояснительной записке использовались следующие термины с соответствующими определениями:

Нейронная сеть – математическая модель, построенная по принципу организации и функционирования биологических нейронных сетей.

Алгоритм слежения – алгоритм обработки видео, позволяющий определять положение выбранного объекта в каждом кадре.

Оперативная память – энергозависимая часть компьютерной памяти, в которой во время работы компьютера хранится выполняемый машинный код, входные, выходные и промежуточные данные, обрабатываемые процессором.

GPS – спутниковая система навигации, определяющая местоположение объектов на Земле.

Обозначения и сокращения

CNN – convolutional neural network, сверточная нейронная сеть.

FPS – frames per second, количество кадров в секунду.

GPS – global positioning system, система глобального позиционирования.

RAM – random access memory, оперативная память.

АС – алгоритм слежения.

Д – диспетчер.

К – камера.

НС – нейронная сеть.

Содержание

Введение

Современные армии стараются делать все от них зависящее, чтобы уменьшить количество потерь личного состава. Для этого солдаты получают самую современную экипировку, средства связи и вооружение. Наземные операции проводятся только в крайних случаях, в основном используются ракетные или бомбовые удары с воздуха. Но зачастую выиграть войну без проведения наземной операции не представляется возможным [1].

Одним из перспективных решений этого вопроса является полная замена солдат роботами. Разработки этого направления ведутся во многих странах. Большинство роботов пока не оснащается летальным оружием, решая такие задачи как обезвреживание мин и фугасов, проведение разведки и наблюдения.

Военный робот состоит из двух компонентов: аппарата, который управляется дистанционно, и пульта, с которого происходит управление. Роботизированные механизмы различаются по степени автономности, они могут в большей или меньшей степени следовать заданной программе и обходиться без постоянного вмешательства со стороны человека. Уже сегодня существует десятки видов военных роботов, которые отличаются габаритными размерами, формой корпуса, шасси, оснащением [2]. Современных наземных военных роботов можно разделить на следующие группы:

1. боевые;
2. разведывательные;
3. инженерные;
4. тыловые.

Последняя категория роботов осуществляет тыловую поддержку пехотных подразделений. Эти роботы способны следовать за отделением и переносить тяжелые грузы, оставляя солдату только рюкзак с самым необходимым и при этом неся на себе более тяжелые вещевые мешки. Еще одной типичной задачей этих роботов является замена машин с экипажами в

опасных задачах, например, доставка боеприпасов на передовую или эвакуация раненых из зоны боевых действий в более безопасный район. В данном случае шагающие роботы являются более предпочтительной категорией, гарантирующим мобильность близкую к мобильности человека. Однако на данное время тяжелые роботы, предназначенные для снабжения, остаются колесными или гусеничными [3].

Для сопровождения солдата имеются несколько режимов управления: ручной и автоматизированный. В ручном способе солдат использует пульт дистанционного управления для подачи команд роботу на движения. Данный способ не позволяет солдату вести наблюдение, так как концентрирует всё его внимание на управлении роботом. Автоматизированный способ лишен данного недостатка [4]. Робот, используя данные с различных датчиков, однозначно идентифицирует солдата, которого необходимо сопровождать (лидера), и автоматически следует за ним на фиксированном расстоянии.

Под объектом исследования понимается использование алгоритма слежения и нейронной сети для слежения за человеком в видеопотоке.

Предметом исследования является обзор различных способы получения информации о текущем положении лидера и предложен алгоритм, который используют только информации с видеокамеры, а также разработано и протестировано программное обеспечение, которое выполняет заданную функцию.

Научная новизна работы заключается в исследовании применимости существующих алгоритмов слежения в военной мобильной робототехники, что выражается в оценке количественных характеристик работы программы. Также проведенный анализ возникших проблем и предложенные решения для их устранения создают почву для дальнейших научных работ.

В ходе исследования возможности использования алгоритма слежения для сопровождения солдата были написаны две научные статьи: Алгоритм слежения по распознанным по видеопотоку объектам и *Development of the video stream object detection algorithm (VSODA) with tracking*.

1 Техническое задание

Для успешного выполнения исследования необходимо до начала определить четкие требования к итоговому результату, что позволит не отвлекаться на второстепенные задачи. В данном разделе поставлены требования к конечному программному обеспечению.

1.1 Основные задачи и цели

Целью работы является разработка программного обеспечения для системы технического зрения, которое позволит следить за человеком и определять его положение в локальных координатах камеры.

Программа будет способна решать следующие задачи:

1) автоматическое распознавание человека на кадре видео;

2) слежение за распознанным человеком в следующих кадрах видео;

3) выдача положения человека в локальных координатах камеры.

1.2 Назначение разрабатываемой программы

Данная программа разрабатывается для применения в тыловых роботах, которые с ее помощью смогут сопровождать солдат в автоматическом режиме. Это позволит солдатам не отвлекаться на управление роботом и полностью сконцентрироваться на выполнении боевой задачи.

1.3 Требования к функциям программы

Основные функции программы представлены в виде нумерованного списка:

1) распознавание человека на изображении и выдача его координат (левая верхняя точка, ширина и высота) в локальных координатах камеры;

2) выдача предполагаемого положения человека в текущем кадре видео на основе его положения в предыдущем кадре;

3) информирование об отсутствии человека в кадре;

4) отображение траектории движения отслеживаемого человека за 20 кадров;

5) вывод количества обрабатываемых кадров видео в секунду.

14 Требования к техническому обеспечению

Для разработки и тестирования программы необходим персональный компьютер или ноутбук с достаточными вычислительными ресурсами для запуска программы. Конкретные количественные характеристики невозможно определить до этапа тестирования программы. Также необходимо камера, которая выдает цветное изображение с 3 каналами (RGB) и разрешением не ниже 640x480.

1.5 Требования к программному обеспечению

Используемые в работе программные инструменты накладывают ограничения на программное обеспечение компьютера. Последний должен иметь операционную систему Linux, желательно с последней доступной версией ядра для уменьшения вероятности несовместимости версий программ.

2 Литературный обзор

2.1 Обзор аналогов

Имеющиеся на данный момент военные роботы по-разному решают задачу автоматического сопровождения лидера. В данном разделе будут представлены описания каждого из способов, а также приведены примеры роботов, использующих данное решение. В конце раздела будет составлена сравнительная таблица по рассмотренным методам.

2.1.1 GPS сигнал

GPS — спутниковая система навигации, позволяющая в любом месте Земли и околоземном космическом пространстве определять местоположение и скорость объектов. Для использования системы необходим только GPS-приёмник, который предназначен для определения географических координат месторасположения приемной антенны в текущий момент времени на основе данных временных задержек прихода радиосигналов.

Приемный модуль GPS ничего не передает, он только получает данные со спутников и определяет свое местонахождение по средствам решения математической задачи. Чтобы устройство смогло определить свое месторасположение, требуется, чтобы прибор увидел не менее чем четыре спутника. Координаты будут тем точнее, чем больше спутников увидит модуль GPS. Скорость работы и чувствительность приемника определяются используемой в устройстве микросхемой. Под скоростью понимают, как быстро прибор способен определить координаты. Под чувствительностью имеют в виду способность приемника определять максимальное количество спутников.

Для применения данного датчика в задаче сопровождения необходимо чтобы он был установлен на роботе и на лидере. Робот, зная своё местоположение и местоположение лидера, способен построить маршрут движения.

Преимуществом данного способы являются необязательный визуальный контакт с лидером, так как сигнал передаётся с помощью радио.

По этой же причине робот может работать в любых условиях освещённости: как в дневное, так и в ночное время.

Основным недостатком из-за чего данный метод не находит широкого применения является неспособность работать в среде с радиопомехами. Сигнал GPS, как и другие радиосигналы, легко заглушить, что и делается во время военных действий. Без этого сигнала робот не способен функционировать в автоматическом режиме. Также лидер должен иметь дополнительное оборудование. Хотя GPS приёмник и мал по размерам, его отсутствие или повреждение приводит к невозможности использования автоматического режима.

2.1.1.1 Northrop Grumman Camel

Одна из компаний, которая применила вышеназванный способ в своей разработке, это Northrop Grumman. Эта компания разработала роботизированный аппарат Camel для обеспечения логистической поддержки пешего патруля. Эта аббревиатура расшифровывается как: Carry-all Modular Equipment Landrover – модульный вездеход для перевозки всего. Система представляет сбой платформу 6x6, поверх колес которой можно одеть резиновые гусеницы при необходимости. Каждое колесо вращается от электродвигателя, на который подается питание от дизель-электрического генератора.

Двигатель работает на дизельном топливе или JP8, а его бак на 13 литров позволяет работать более 20 часов; подобное решение при приближении к потенциальной опасности позволяет двигаться в бесшумном режиме. Максимальная скорость аппарата составляет 8 км/ч, он может преодолевать уклоны 40%, боковые уклоны 20%, препятствия и брод 0,3 метра. Его груз, уложенный внутри установленной на шасси трубчатой конструкции, может превышать 350 кг [5]. Внешний вид данного робота представлен на рис. 1.

Рисунок 1 – Внешний вид Northrop Grumman Camel

Camel оборудуется сенсорным комплектом обнаружения и обхода препятствий. Им можно управлять в режиме «следуй за мной» или через кабель. В транспортной колонне аппараты могут следовать друг за другом подобно вагонам поезда. Робот может доукомплектовываться опциональными системами, такими как:

1. жесткий кабель;
2. оптоволокно;
3. взаимозаменяемые каналы связи;
4. внешний блок аккумуляторов для увеличения запаса хода;
5. радиочастотные системы.

2.1.1.2 HDT Global Protector

Вторая компания, использовавшая GPS сигнал для автоматического сопровождения солдата, это HDT Global. Гусеничный робот Protector разработан в качестве многозадачной системы обеспечения солдат в полевых условиях. На аппарате установлен дизельный двигатель мощностью 32 л.с., он может перевозить груз массой 340 кг плюс тянуть за собой на прицепе еще 225 кг. Protector может быть быстро разобран на переносные модули, чтобы можно было преодолевать непредвиденные препятствия. Топливный бак на 57 литров позволяет иметь запас хода 100 км. Максимальная скорость робота составляет

8 км/ч. Базовый аппарат управляется дистанционно, а режим круиз-контроля позволяет снизить рабочую нагрузку на оператора [6].

Для вспомогательного оборудования система имеет выход гидравлической системы и разъем с выходной электрической мощностью 2 кВт. Внешний вид робота представлен на рис. 2.

Рисунок 2 – Внешний вид HDT Global Protector

2.1.2 Лидар сигнал

Лидар — технология получения и обработки информации об удалённых объектах с помощью активных оптических систем, использующих явления поглощения и рассеяния света в оптически прозрачных средах.

Принцип действия лидара не имеет больших отличий от радара: направленный луч источника излучения отражается от целей, возвращается к источнику и улавливается высокочувствительным приёмником, в случае лидара — светочувствительным полупроводниковым прибором. Время отклика прямо пропорционально расстоянию до цели. Кроме импульсного метода измерения дистанции применяется фазовый, основанный на определении разности посылаемых и принимаемых фаз модулированных сигналов.

В отличие от радиоволн, эффективно отражающихся только от достаточно крупных металлических целей, световые волны подвержены рассеянию в любых средах, в том числе в воздухе, поэтому возможно не только определять расстояние до непрозрачных (отражающих свет)

дискретных целей, но и фиксировать интенсивность рассеивания света в прозрачных средах. Возвращающийся отражённый сигнал проходит через ту же рассеивающую среду, что и луч от источника, подвергается вторичному рассеиванию. Восстановление действительных параметров распределённой оптической среды является достаточно сложная задача, решаемая как аналитическими, так и эвристическими методами.

При использовании данного датчика для решения задачи автоматического сопровождения лидер носит отражательный маркер. Отраженный от маркера сигнал отличается от других, что позволяет однозначно идентифицировать лидера и определить расстояние до него.

По сравнению с GPS сигналом данный способ определения лидера лишен недостатка невозможностью работы в условии радиопомех. Хотя датчик тоже использует радиосигнал, рабочая частота сигнала отличается от подавляемой. Условия освещенности не оказывают большого влияния на работу робота, что является плюсом.

Для применения данного метода лидер также должен иметь дополнительное оборудование, а именно отражательный маркер. Но это не так критично, как обязательный визуальный контакт лидера и робота. Лидар использует направленный луч, что накладывает ограничения на область применения.

2.1.2.1 Boston Dynamics BigDog

Вышерассмотренный способ нашёл своё применение в роботе BigDog. Это четырёхногий робот с адаптивным управлением, созданный в 2005 году фирмой Boston Dynamics совместно с Foster-Miller, Лабораторией реактивного движения (NASA) и Harvard University Concord Field Station [7].

Робот следует за лидером без прямого управления и GPS. Лидер носит отражательный маркер. Для определения лидера и генерации сигнала управления используется SICK LIDAR. BigDog следует на фиксированном расстоянии. Помимо автоматического режима существует пульт ручного

управления с помощью радиосигнала. Схема работы системы представлена на рис. 3.

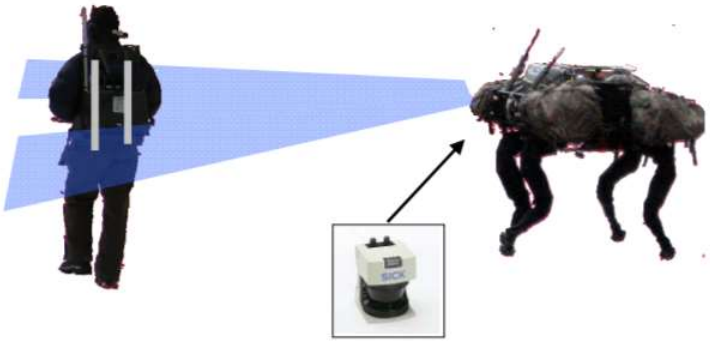

Рисунок 3 – Схема работы системы автоматического сопровождения Big Dog

2.1.3 Алгоритм слежения по видеопотоку

На данный момент существует большое количество open-source алгоритмов, которые по полученному видео потоку и заданному в начальном кадре объекте слежения, могут осуществлять слежение за ним. Под слежением понимается устойчивое определение данного объекта на следующих кадрах и его выделение. Выделяют объект на кадрах чаще всего рамкой контрастного цвета (красный, синий).

Данные алгоритмы имеют подробную документацию в виде научных статей и онлайн учебников, а также примеры работы, что позволяет быстро их освоить приспособить для выполнения конкретной задачи, имеющую свою специфику. Примером реализации таких алгоритмов в виде программного кода может являться открытая платформа OpenCV (Open Computer Vision, открытое компьютерное зрение), в которой имеются такие алгоритмы слежения, как: Meanshift, Camshift, MIL, KCF, TLD. Данное программное обеспечение распространяется по лицензиям GPL и BSD, что позволяет использовать их в коммерческих разработках.

В начальный момент времени указывается отслеживаемый объект путём выделения области на первом полученном кадре с видеокамеры. После чего изображения с видеокамеры обрабатываются для определения относительного положения объекта. В качестве алгоритма поиска объекта

используется метод сдвига среднего значения. Данный алгоритм обладает большой скоростью работы. По этим данным вырабатываются управляющие сигналы на двигатели робота.

Пока не существуют успешных конкурентоспособных проектов, использующих данный способ для автоматического сопровождения солдата. В данной работе будет рассмотрен вопросы возможности применения данных алгоритмов в реальной работе и вероятные проблемы при использовании данного способа2.

Преимуществом данного метода по сравнению с лидаром это отсутствие необходимости лидера иметь дополнительное оборудование. Алгоритм использует только информацию с видеопотока. Чем контрастнее цель, тем выше качество слежения, но в условии военных действий это не часто возможно.

Самый большой недостаток — это зависимость от условий освещения. Видеокамера работает, получая отраженный с объектов свет. Если света мало, то изображение с камеры будет малоинформативным.

2.1.4 Комбинация лидара и алгоритма слежения

Данный способ позволяет получить полную 3D модель солдата. Далее робот, сканируя окружающую местность, пытается найти похожий объект и следует за ним. Этот метод является наиболее технологичным, который комбинирует преимущества обоих вышеназванных способов. В тоже время для использования требуется наличия большего числа датчиков и высоких вычислительных ресурсов, так как строится полная 3D модель окружающего мира. Тем не менее одна из существующих систем применяет данную технологию.

2.1.4.1 Lockheed Martin SMSS

Роботизированная система поддержки отделения SMSS, разработанная компанией Lockheed Martin, представляет собой наземный робот типа «мула», который был проверен в реальных боевых условиях. Аббревиатура SMSS расшифровывается как: Squad Mission Support System, система поддержки

отряда. Система была выбрана в 2011 году американской армией для своих испытаний, в следствие чего четыре аппарата SMSS были развернуты в войсках в 2012 году. Роботы прошли испытания, так как их способность нести на себе почти 700 кг при самостоятельном движении за солдатами оказалась чрезвычайно полезной. В одном случае на робота было нагружено более одной тонны различных запасов и при этом он работал безошибочно.

Разработанный примерно в 2005 году и постоянно модернизирующийся робот SMSS базируется на аппарате Land Tamer 6x6 XHD от PFM Manufacturing Inc, изготовленном из алюминия для морских судов с турбодизельным двигателем мощностью 80 л.с. Некоторые из предоставленных характеристик варианта Block 1: общая масса 1955 кг, грузоподъемность 682 кг, аппарат может перевозиться внутри вертолетов CH-53 и CH-47 или на подвесе UH-60 [8].

Компания Lockheed Martin сосредоточилась на добавлении автономных возможностей, SMSS способен работать в различных режимах, например ручное управление, дистанционное управление, голосовые команды, возврат к оператору, движение к месту по выбранным точкам координат, возвращение по сформированной траектории, навигация по координатным точкам GPS, следование за человеком и следование за транспортным средством. Внешний вид робота представлен на рис. 4.

Рисунок 4 – Внешний вид Lockheed Martin SMSS

2.1.5 Сравнение решений

По всем рассмотренным способам была составлена сводная таблица 1, в которой в строках содержатся возможные достоинства метода. По вертикали отмечены все способы автоматического сопровождения.

Таблица 1. Сравнительный анализ решений

Название характеристики	**GPS сигнал**	**Лидар сигнал**	**Алгоритм слежения**	**Комбинация**
Необязательность визуального контакта	Да	Нет	Нет	Нет
Независимость от освещения	Да	Да	Нет	Нет
Устойчивость в условиях помех	Нет	Да	Да	Да
Отсутствие необходимости лидера иметь дополнительное оборудование	Нет	Нет	Да	Да

Как видно из таблицы, ни один из методов не обладает всеми преимуществами. Это означает, что при выборе решения для автоматического сопровождения нужно учитывать специфику применения робота.

Хотя комбинированный метод и предлагает повышенное качество слежения за счёт наличия данных с нескольких типов датчиков, интерес представляет функционирования системы сопровождения, основанной только на алгоритме слежения по видеопотоку. Данная разработка позволит не только снизить стоимость конечного робота, но и позволит функционировать роботам, использующим комбинированный способ, в случае отказа лидара.

2.2 Обзор алгоритмов слежения

Обнаруживать объект на видео возможно двумя способами: распознаванием и слежением. В случае распознавания программе известно, как выглядит объект слежения, и она последовательно проверяет части изображения с целью найти похожие объекты. Недостатком такого подхода

является невозможность следить за объектом в случае его частичного или полного перекрытия, а также, если внешний вид объекта слежения сильно меняется.

Для слежения необходимо в начальный момент времени выделить объект, далее программа будет отслеживать объект, оценивая оптический поток, который характеризует относительное изменение положение объектов на следующем кадре видео. Так как известно где находился объект в предыдущих кадрах, то известна скорость и направление движения объекта.

Используя эти данные возможно с большой точностью предсказать следующее положение объекта. В случае долговременной потери объекта слежения из поля зрения объектива программа уже не сможет работать.

Исходя из вышеназванных недостатков двух способов в современных алгоритмах слежения, используется комбинированный способ. В начальный момент времени выделяется объект слежения, и затем классификатор обучается по данному изображению объекта и по следующим, которые были получены алгоритмом слежения по относительному смещению объекта слежения. В итоге классификатор обучается на конкретном объекте слежения, что позволяет добиться лучшего распознавания в данных условиях. Данный способ требует постепенного изменения внешнего вида и не способен отрабатывать резкое изменение освещения. Но комбинированный способ всё равно требует ручного выделения объекта в начальный момент времени и в случае долговременной потери объекта.

Для данного исследования был составлен обзор алгоритмов слежения, использующих вышеназванный способ. Ниже приведена краткое описание каждого из алгоритмов, а также их достоинства и недостатки.

2.2.1 MIL

MIL расшифровывается как Multiple Instance Learning, обучение по множественным примерам. Данный алгоритм использует текущее положение объекта как положительный пример для обучения классификатора, а также несколько частей изображения, которые по размеру равны объекту и

находятся в небольшой окрестности рядом с текущем положением объекта, как потенциально положительные примеры. Таким образом, создаётся набор положительных примеров. Таким образом, если текущее положение объекта слежения неточно, то существует шанс того, что положительный пример с верным текущем положением объекта находится в наборе. Устойчиво работает при частичном перекрытии объекта, но при долговременном полном перекрытии алгоритм теряет объект слежения [9].

В частности, подавая в алгоритм тренировочные экземпляры $\{(X_1, y_1),...,(X_n, y_n)\}$ в текущем кадре, где набор $X_i = \{x_{i1},...,x_{im}\}$ и его метка $y_i = \{0,1\}$, и набор из M кандидатов на слабые классификаторы $H = \{h_1, h_2,..., h_M\}$, MIL последовательно выбирает K слабых классификаторов из набора кандидатов с помощью следующего критерия:

$$h_k = argmax\ L(H_{k-1} + h),$$

где $L = \sum_i (y_i \cdot \log p_i + (1 - y_i) \cdot \log(1 - p_i))$ - логарифмическое правдоподобие наборов; $H_{k-1} = \sum_{i=1}^{k-1} h_i$ - сильный классификаторов, состоящий из первых $k-1$ слабых классификаторов.

Стадии работы алгоритмы представлены на рис. 5.

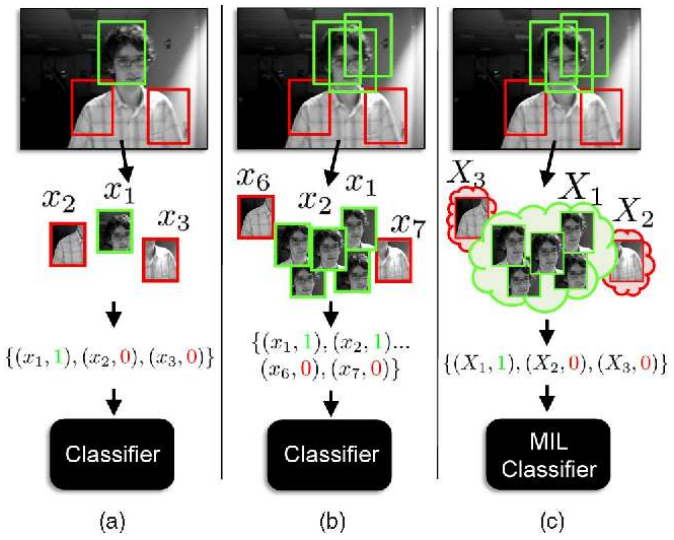

Рисунок 5 - Стадии работы MIL

2.2.2 TLD

Аббревиатура TLD расшифровывается как: Tracking, Learning, Detection или Слежение, Обучение, Распознавание. Исходя из названия, можно определить, что данный алгоритм декомпозирует долговременное слежение в три компоненты: кратковременное слежение, обучение и распознавание [10].

Модуль слежения покадрово сопровождает объект. Модуль распознавания корректирует модуль слежения, если это необходимо. Модуль обучения аккумулирует все изображения объекта и стремится снизить ошибку. Схема работы алгоритма представлена на рисунке 6.

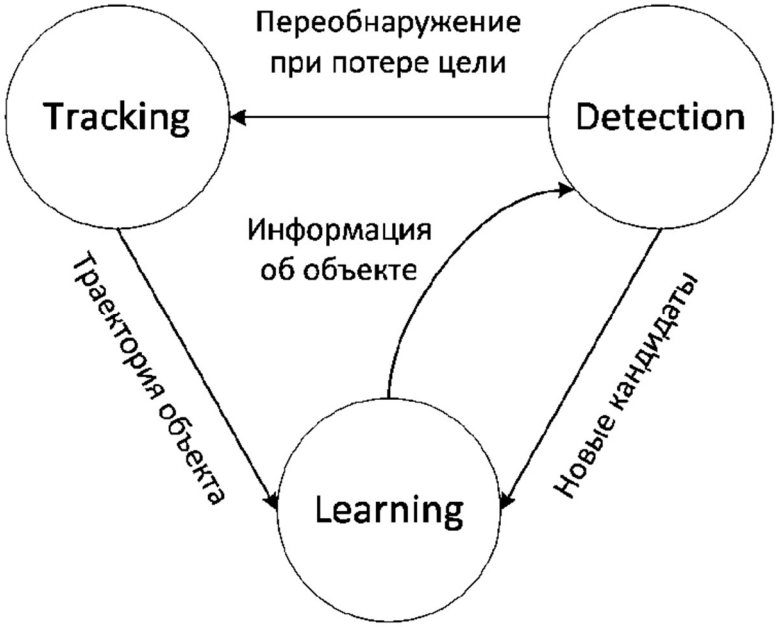

Рисунок 6 – Схема работа алгоритма TLD

Принцип работы заключаются в том, что на каждом кадре:

1) находятся кандидаты;

2) определяется траектория;

3) все кандидаты в окрестности траектории считаются положительными примерами;

4) все остальные кандидаты считаются отрицательными;

5) на основе P-N constraints модифицируется классификатор.

Одним из важнейших параметров алгоритма является зона поиска объекта. Она ограничена следующим выражением:

$$numWindows = \sum_{s \in 1.2^a} [\frac{n - s \cdot (\omega + d_x)}{s \cdot d_x} \cdot \frac{m - s \cdot (h + d_y)}{s \cdot d_y}],$$

где $s \in 1.2^a$; $a \in -maxScale...minScale$; $maxScale$ - максимальный масштаб; $minScale$ - минимальный масштаб; n - ширина изображения; m - высота изображения; ω - ширина изначальной ограничивающей рамки объекта; h - высота изначальной ограничивающей рамки объекта; d_x и d_y - поля между двумя соседними окнами поиска и устанавливаются как $\frac{1}{10}$ от величины изначальной ограничивающей рамки.

Среди достоинств данного подхода стоит отметить:

1) работа в реальном времени;

2) неограниченная длительность слежения;

3) отсутствие стадии предобучения - не требуется априорная информация об объекте;

4) стабильность к кратковременным перекрытиям объекта слежения;

5) способен определять изменения масштаба объекта слежения.

Немаловажно будет упомянуть и о недостатках данного алгоритма:

1) слежение только за одним объектом;

2) требуется ручная инициализация цели.

2.2.3 GOTURN

Данный алгоритм основан на применении сверточных нейронных сетей. Изначально нейронная сеть уже обучена на большом классе объектов, полученных с ImageNet. На каждом кадре видео нейронная сеть стремится найти выбранный в начальный момент времени объект. Причем объект слежения должен принадлежать к классам объектов, которые нейронная сеть способна распознать.

К достоинствам данного алгоритма стоит отнести верность определения потери объекта. К недостаткам нужно отнести низкую

устойчивость слежения в случаях заслонения объекта, отсутствие подстройки под конкретный объект слежения. Общий принцип работы представлен на рисунке 7.

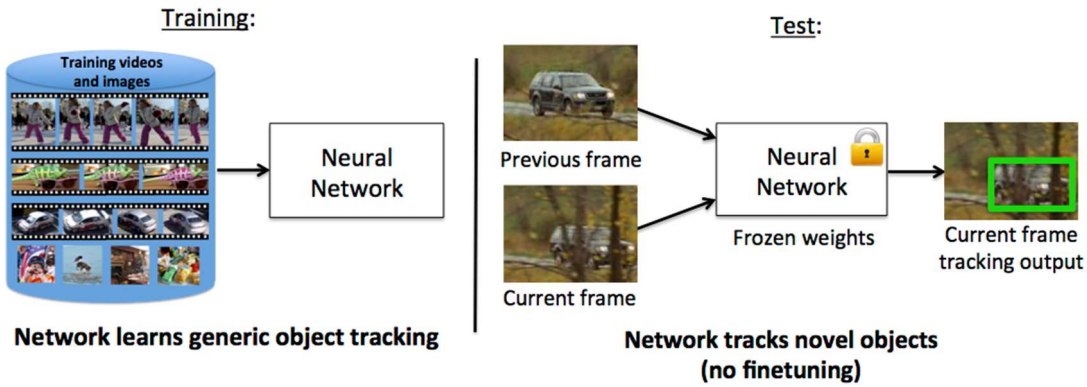

Рисунок 7 – Принцип работы алгоритма GOTURN

2.3 Обзор нейронных сетей

Главным недостатком алгоритмов слежения является ручной перезапуск программы в случае долговременной потери объекта, так как алгоритм самостоятельно уже не способен найти объект. Одним из способов решить данную проблему – это использовать искусственные нейронные сети (ИНС) для распознавания человека в кадре. Это позволит повысить качество слежения. В данной главе рассмотрены различные архитектуры ИНС, приведены их достоинства и недостатки в контексте решения нашей задачи.

Искусственная нейронная сеть — математическая модель, а также её программное или аппаратное воплощение, построенная по принципу организации и функционирования биологических нейронных сетей — сетей нервных клеток живого организма. Состоит нейронная сеть из элементарных единиц – нейронов [11].

Нейрон — это вычислительная единица, которая получает информацию, производит над ней простые вычисления и передает ее дальше. Они делятся на три основных типа, которые представлена на рис. 8:

1. входной (синий);
2. скрытый (красный);
3. выходной (зеленый).

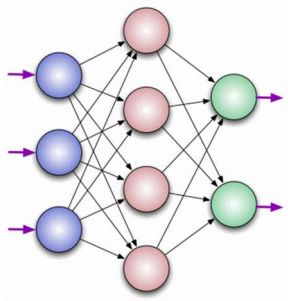

Рисунок 8 – Модель ИНС

Входной слой получает информацию, затем *n* скрытых слоев ее обрабатывают, и выходной слой выводит результат.

В настоящее время нейронные сети нашли широкое применение для решения задач распознавания образов на изображениях, анализа голоса и текста, прогнозирования ситуации на биржевых рынках и т.д. Их популярность обусловлена высокими показателями качества работы хорошо обученной нейронной сети, что связано с:

1. созданием обширной методологической базы по обучению сетей;
2. ростом вычислительных мощностей;
3. накоплением большого количество обучающих данных [12].

На данный момент существует большое количество архитектур нейронных сетей, предназначенных для решения различных задач. Рассмотрим наиболее распространенные архитектуры.

2.3.1 Сети прямого распространения

Данные нейронные сети очень просты — они передают информацию от входа к выходу. Простейшая рабочая сеть состоит из двух входных и одного выходного нейрона и может моделировать логический вентиль — базовый элемент цифровой схемы, выполняющий элементарную логическую операцию.

Впервые рабочий прототип прямой сети был представлен Розенблатом в 1958 году и мог распознавать буквы на изображении с помощью фотодатчиков. Сети прямого распространения (Feed Forward Neural Network, FFNN) обычно обучают методом обратного распространения ошибки, подавая модели на вход пары входных и ожидаемых выходных данных. Под ошибкой

обычно понимаются различные степени отклонения выходных данных от исходных (например, среднеквадратичное отклонение или сумма модулей разностей).

При условии, что сеть обладает достаточным количеством скрытых нейронов, теоретически она всегда сможет установить связь между входными и выходными данными. На практике использование сетей прямого распространения ограничено, и чаще они используются совместно с другими сетями. Схема FFNN представлена на рис. 9.

Рисунок 9 – Схемы FFNN

2.3.2 Рекуррентные нейронные сети

Данный вид сетей имеет строение сходное с сетями прямого распространения, но со смещением во времени (рисунок 5): нейроны получают информацию не только от предыдущего слоя, но и от самих себя в результате предыдущего прохода. Следовательно, здесь важен порядок, в котором мы подаем информацию и обучаем сеть.

У рекуррентных нейронных сетей (Recurrent Neural Network, RNN) есть одна большая проблема — это проблема исчезающего (или взрывного) градиента: в зависимости от используемой функции активации информация со временем теряется так же, как и в очень глубоких сетях прямого распространения. Казалось бы, это не такая уж серьезная проблема, так как это касается только весов, а не состояний нейронов, но именно в весах хранится информация о прошлом. Если вес достигнет значения 0 или 1 000 000, то информация о прошлом состоянии станет не слишком информативной [13].

RNN могут использоваться в самых разнообразных областях, так как даже данные, не связанные с течением времени (не звук или видео) могут быть представлены в виде последовательности. Картинка или строка текста могут

подаваться на вход по одному пикселю или символу, так что вес будет использоваться для предыдущего элемента последовательности, а не для того, что случилось X секунд назад. В общем случае, рекуррентные сети хороши для продолжения или дополнения информации, например, автодополнения. Схема FFNN представлена на рис. 10.

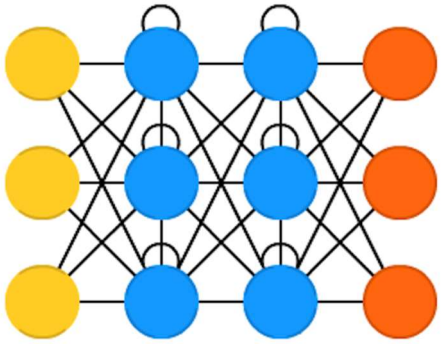

Рисунок 10 – Схема RNN

2.3.3 Сверточные нейронные сети

Сверточные нейронные сети (Convolutional Neural Networks, CNN) кардинально отличаются от других сетей. Они используются в основном для обработки изображений, иногда для аудио и других видов входных данных. Впервые данная архитектура была предложена французским ученым Яном Лекуном. Типичным способом применения CNN является классификация изображений: если на вход подается изображение кошки, сеть выдаст «кошка», если картинка собаки — «собака» [14].

Такие сети обычно используют «сканер», не обрабатывающий все данные за один раз. Если есть изображение 200x200, то в случае другой НС будет необходимо создать слой сети из 40 тысяч узлов. Вместо этого сверточная сеть считает квадрат размера 20x20 (обычно из левого верхнего угла), затем сдвинется на 1 пиксель и считает новый квадрат, и т.д. Изображение не разбивается на квадраты, а скорее заданный квадрат передвигается по изображению.

Эти входные данные затем передаются через сверточные слои, в которых не все узлы соединены между собой. Вместо этого каждый узел соединен только со своими ближайшими соседями. Эти слои имеют свойство

сжиматься с глубиной, причём обычно они уменьшаются на какой-нибудь из делителей количества входных данных. Например, 20 узлов в следующем слое превратятся в 10, в следующем — в 5, часто используются степени двойки.

Кроме сверточных слоев есть также так называемые слои объединения (pooling layers). Объединение — это способ уменьшить размерность получаемых данных, например, из квадрата 2x2 выбирается и передается наиболее красный пиксель. На практике к концу CNN прикрепляют FFNN для дальнейшей обработки данных. Такие сети называются глубокими (DCNN), но названия их обычно взаимозаменяемы. На рисунке 11 представлена типичная схема глубокой сверточной сети.

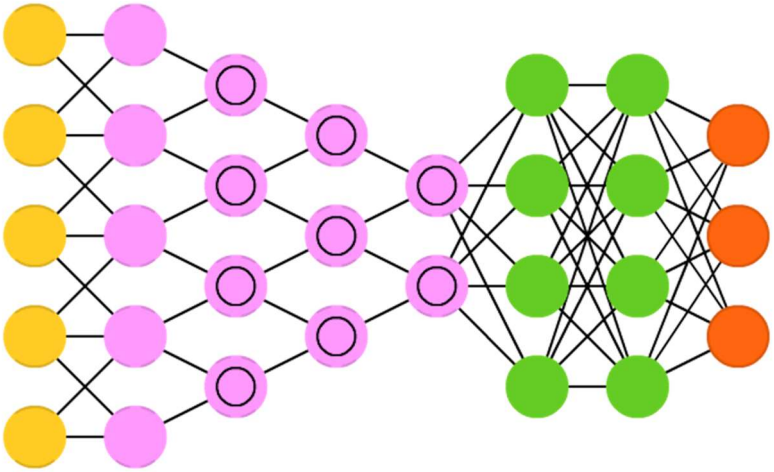

Рисунок 11 – Схема DCNN

В обычном перцептроне, который представляет собой полносвязную нейронную сеть, каждый нейрон связан со всеми нейронами предыдущего слоя, причем каждая связь имеет свой собственный весовой коэффициент. В сверточной нейронной сети в операции свёртки используется лишь ограниченная матрица весов небольшого размера, которая перемещается по всему изображению, формируя после каждого сдвига сигнал активации для нейрона следующего слоя с аналогичной позицией.

Для различных нейронов выходного слоя используются одна и та же матрица весов, которую также называют ядром свёртки. Её интерпретируют как графическое кодирование какого-либо признака, например, наличие наклонной линии под определенным углом. Тогда следующий слой,

получившийся в результате операции свёртки такой матрицей весов, показывает наличие данного признака в обрабатываемом слое и её координаты, формируя так карту признаков (англ. feature map).

В сверточной нейронной сети набор весов не один, а некоторая совокупность, кодирующая элементы изображения (например, линии и дуги под разными углами). При этом такие ядра свертки не определяются заранее, а формируются самостоятельно путём обучения сети. Проход каждым набором весов формирует свой собственный экземпляр карты признаков, делая нейронную сеть многоканальной (много независимых карт признаков на одном слое).

Следует отметить, что при переборе слоя матрицей весов её передвигают обычно не на полный шаг (размер этой матрицы), а на небольшое расстояние. Так, например, при размерности матрицы весов 5×5 её сдвигают на один или два нейрона (пикселя) вместо пяти, чтобы не пройти мимо нужного признака.

Операция субдискретизации (англ. subsampling, англ. pooling, также «операция подвыборки» или операция объединения), выполняет уменьшение размерности сформированных карт признаков. В данной архитектуре сети считается, что информация о факте наличия искомого признака важнее точного знания его координат, поэтому из нескольких соседних нейронов карты признаков выбирается максимальный и принимается за один нейрон уплотнённой карты признаков меньшей размерности. За счёт данной операции, помимо ускорения дальнейших вычислений, сеть становится более инвариантной к масштабу входного изображения.

После нескольких прохождений свёртки изображения и уплотнения с помощью пулинга система перестраивается от конкретной сетки пикселей с высоким разрешением к более абстрактным картам признаков, при этом на каждом следующем слое увеличивается число каналов и уменьшается размерность изображения в каждом канале. В конечном итоге остаётся большой набор каналов, хранящих небольшое число данных, которые

интерпретируются как самые абстрактные понятия, выявленные из исходного изображения.

Эти данные объединяются и передаются на обычную полносвязную нейронную сеть, которая тоже может состоять из нескольких слоёв. При этом полносвязные слои уже утрачивают пространственную структуру пикселей и обладают сравнительно небольшой размерностью (по отношению к количеству пикселей исходного изображения). На рисунке 12 представлен принцип работы сверточной нейронной сети в визуальной форме.

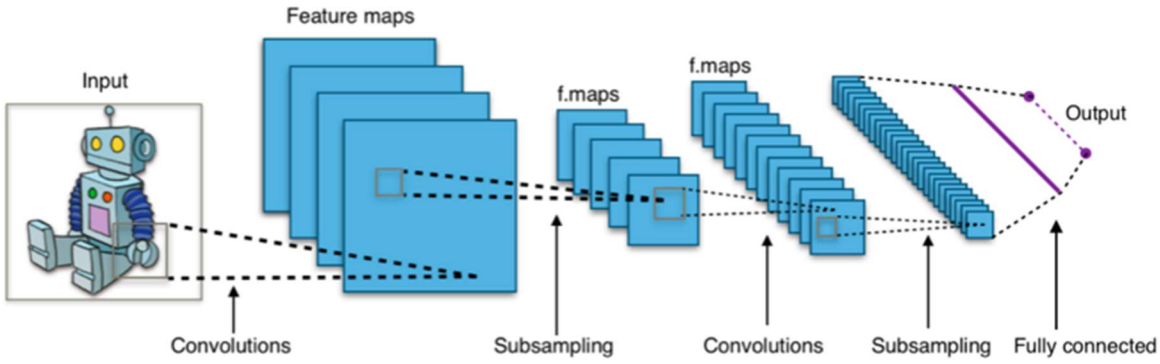

Рисунок 12 – Принцип работы CNN

К достоинствам сверточных нейронных сетей можно отнести:

1. меньший по сравнению с сетями прямого распространения набор настраиваемых весов;

2. возможность использования параллельных вычислений, в том числе на видеокартах, что ускоряет процесс обучения и работы нейронной сети;

3. устойчивость к искажениям изображений, таким как поворот или сдвиг;

4. возможность использования классических алгоритмов обучения;

5. высокие показатели точности при классификации изображений [15].

Главным недостатком данной архитектуры является отсутствие четкой методики выбора структуры сверточной нейронной сети и ее параметров (размерность ядра свертки, количество слоев, функции активации и т.д.). В связи с этим разработка архитектуры происходит эмпирическим путем на основе предыдущего опыта использования данного вида нейронных сетей и некоторых обобщенных рекомендаций.

3 Разработка и тестирование программы

Оценив существующие алгоритмы слежения, было решено выбрать в качестве базового алгоритм TLD. Он имеет лучшие показатели в ситуациях, которые свойственны в задаче сопровождения. А именно частое перекрытие объекта слежения, кратковременная потеря объекта, изменение масштаба объекта.

Для классификации объекта, а также определения его положения на кадре было решено использовать сверточную нейронную сеть. Данный вид нейросетей позволяет получить наилучшие результаты при распознании изображений. Кроме того, сверточная нейронная сеть имеет меньше настраиваемых параметров, чем аналогичные по точности архитектуры.

На вход сети подается изображение с тремя цветовыми каналами (RGB). На выходе у сети пять параметров: класс объекта, координаты верхнего левого угла, ширина и высота захватывающей рамки (bounding box) [16].

Принцип работы алгоритма следующий. Начальный кадр видео обрабатывается нейронной сетью, которая классифицирует объекты на кадре и определяет положение человека. Далее эта информация поступает на алгоритм TLD, который осуществляет слежение за объектом. Через небольшие интервалы времени сравниваются положения объекта, полученные при помощи алгоритма TLD и нейронной сети. Если различие превышает порог, то алгоритму TLD указывается новое положение объекта слежения, которое было определено нейронной сетью. НС в данном случае играет роль эксперта.

В данном разделе описаны шаги по созданию программного обеспечения, выполняющего данную задачу, а также его тестированию с оценкой количественных и качественных показателей.

3.1 Архитектура программы

Первым шагом в создании программы является планирование ее архитектуры. Поскольку уже имеются два компоненты – алгоритм слежения и нейронная сеть, то их решено было отделить в отдельные модули.

Поскольку оба компонента работают с камерой, а именно получают кадр видео для обработки и камера одна, то всю логику работы с камерой предпочтительней убрать в один модуль и создать простой интерфейс для получения текущего кадра. Кроме того, в модуль камеры вынесены такие операции, как:

1. вывод изображения;
2. расчёт FPS;
3. вывод отладочной информации в консоль.

Данные компоненты находятся по сути на одном уровне абстракции и их использовать напрямую не является хорошей практикой. Это приведёт к сложностям в дальнейшей отладке и добавлению нового функционала. По этим причинам был организован компонент «Диспетчер», который работает с данными компонентами. Это позволяет убрать всю логику взаимодействия АС и НС в «Диспетчер» и оставить в главной программе только настройку «Диспетчера» и его работу. Модули программы, а также их связи представлены на рис. 13.

Рисунок 13 - Модули системы

Блок-схемы работы отдельных функций модулей, а также их взаимодействие между собой приведено в приложениях.

3.2 Используемые программные инструменты

На данный момент не имеет смысла программировать алгоритм слежения и обучать нейронную сеть, так как уже имеются готовые реализации, которые уже оптимизированы и относятся к открытому программному обеспечению. Это позволяет сэкономить время и сконцентрироваться на

других аспектах работы. Далее приведен список используемых программных инструментов с их кратким описанием.

3.2.1 Библиотека компьютерного зрения OpenCV

OpenCV (Open Source Computer Vision Library) выпускается под лицензией BSD и, следовательно, бесплатна как для академического, так и для коммерческого использования. Данная библиотека имеет интерфейсы C ++, Python и Java и поддерживает Windows, Linux, Mac OS, iOS и Android. OpenCV был разработан для вычислительной эффективности и с большим вниманием к приложениям реального времени. Написанная в оптимизированном C/C ++, библиотека может использовать многоядерную обработку [17].

OpenCV насчитывает более 47 тысяч пользователей сообщества пользователей и приблизительное количество загрузок, превышающих 14 миллионов. Использование варьируется от интерактивного искусства до робототехники [18].

В данной работе используется интерфейс Python для работы с OpenCV. Это связано с выбором НС, которому посвящена следующая глава.

3.2.2 Система обнаружения объектов YOLO

В качестве НС была выбрана YOLO (You only look once). Это сверточная нейронная сеть, которая одновременно находит множество объектов на кадре, показывает их положение, класс и уверенность в распознавании. YOLO тренируется на всем изображении и поэтому оптимизирует обнаружение объектов, не используя традиционные подходы к поиску объектов на кадре [19]. Архитектура сети представлена на рис. 14.

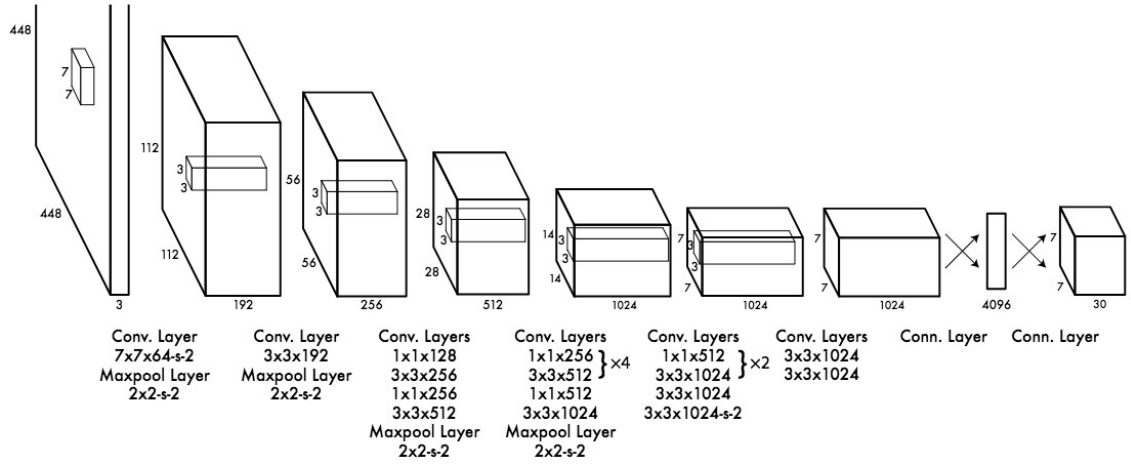

Рисунок 14 - Архитектура сети YOLO [20]

3.3 Материал для тестирования

Для тестирования программы были записаны видео в лесной местности, которые были разделены на несколько категорий, которые представлены ниже.

1. Заграждение человека предметом, за которым можно все равно распознать объект слежения. В лесу этим предметом являются кусты.

2. Заграждение человека предметом, за которым нельзя распознать объект слежения. Эту роль хорошо выполняют деревья

3. Комбинация первого и второго случаев.

4. Прямолинейное движение человека по тропинке.

Эти ситуации являются типовыми и позволяют оценить алгоритм с разных углов. Тестовые видео были записаны на камеру мобильного телефона. Разрешение видео составляет 640x480.

3.4 Измерение затрат ресурсов и качества слежения

Для измерения затрат вычислительных ресурсов был взят ноутбук со следующими характеристиками:

1. Операционная система: Ubuntu MATE 16.04.3 LTS (Xenial Xerus) (Oracle VM для Windows, версия 5.1.24 r117012 (Qt5.6.2))

2. Компилятор: GCC (Ubuntu 5.4.0-6ubuntu1~16.04.5) 5.4.0 20160609

3. Версия OpenCV: 3.3.1

4. Процессор: Intel Core i7 6500U 2 cores

5. Оперативная память: 4 GB

Ноутбук не полностью поддерживал Linux, поэтому тестирование проводилось на виртуальной машине, которой было выделено половина общих вычислительных ресурсов. Далее приведен список метрик, по которому измерялись затраты ресурсов и их показатели в общем по тестовым видео или для каждого из них.

3.4.1 Количество кадров в секунду

Количество обработанных кадров в секунду (FPS) напрямую связано с вычислительной сложностью алгоритма. Если данный показатель слишком мал, то робот не будет способен быстро реагировать на изменение обстановки и потеряет лидера. Хорошим показателем является 20 кадров в секунду или выше.

Для всех тестовых видео FPS держался в интервале 3-5. Это показывает, что алгоритм вычислительно сложный и его пока невозможно применить в реальном роботе.

3.4.2 Использование процессора и оперативной памяти

Загрузка процессора и объем занятой оперативной памяти помогают определить требования к аппаратной части робота. Чем производительнее установленное оборудование, тем больше электроэнергии оно требует для своей работы, что негативно сказывается на времени автономной работы.

Для измерения данных показателей использовалась команда top. Она среди прочих показателей выводит %CPU и %MEM.

%CPU – это использование процессора. Доля использования процессора, используемая программой. Выражается в процентах. По умолчанию показывает для одного ядра процессора. Если система многоядерная, то доля может быть больше 100%. Для примера если имеются 3 ядра, которые используются на 60%, то показатель %CPU покажет 180%.

%MEM – это использование оперативной памяти. Выражается в процентах от общего размера оперативной памяти.

Для более точного определения данных метрик было проведено несколько измерений, результаты которых приведены в таблице 2.

Таблица 2. Использование процессора и оперативной памяти

Номер замера	**Значение %CPU**		**Значение %MEM**	
	2 ядра, %	**1 ядро, %**	**%**	**МБ**
Загрузка сети	97.0	48.5	18.5	757.6
1	176.7	88.4	30.7	1257.47
2	147.2	73.6	32.0	1310.72
3	153.5	76.8	30.4	1245.18
4	167.1	83.6	30.4	1245.18
5	184.1	92.1	30.5	1249.28
6	157.5	78.8	30.5	1249.28
Среднее	164.35	82.21	30.75	1259.51

Итоговая средняя величина использования процессора и оперативной памяти ещё раз показывает, что алгоритм предъявляет высокие требования к аппаратной части программы. В следующем разделе проведена профилировка программа с целью узнать какие именно части программы потребляют наибольшее количество ресурсов.

3.4.3 Профилировка программы

Профилировка программы помогает понять какие части программы наиболее часто вызываются, а также узнать какую долю времени от общего происходит их выполнение. Это позволяет сконцентрироваться на оптимизации критически важных частей программы.

В качестве профилировщика использовался cProfile. Для запуска программы вместе с профилировщиком нужно написать в терминале следующую команду:

$python\ -m\ cProfile\ -o\ output.pstats\ script.py\ arg1\ arg2$, где:

1) *python* - интерпретатор Python. В данном случае явно вызывается 3 версия;

2) *−m* - ключ, показывающий, что дальше будет указан модуль Python, которой необходимо запустить;

3) *cProfile* - название модуля-профилировщика;

4) *−o* - ключ, указывающий, что дальше будет указано имя выходного файла, в котором будет собрана вся информация;

5) *output.pstats* - название выходного файла;

6) *script.py* - название программы;

7) *arg*1 *arg*2 - входные аргументы программы.

В нашем случае команда выглядела следующим образом:

*python*3 *−m cProfile −o profile.pstats main.py −−version* 320

−−image show −−calibration on −−bbox coord −−debug gui

−−net on −−source cam

Чтобы получить собранную профилировщиком информацию были запущены следующие команды:

1) *python*3 *−m pstats profile.pstats* - открытие файла с информацией. Данная информация имеет табличный вид.

2) *sort time* - сортировка таблицы по общему времени работы функции.

3) *stats* 10 - вывод только первых десяти строк таблицы.

Данный список представлен в таблице 3. Все названия столбцов сохранения в исходном виде, ниже приведены расшифровка этих названий:

1) *ncalls* – общее число вызовов;

2) *tottime* – общее время, потраченное на выполнение данной функции. Не учитывается вызов подфункций;

3) *percall – tottime* деленное на *ncalls*. Показывает среднее время на один вызов функции;

4) *cumtime* – кумулятивное время, которое потрачено в этой функции и во всех подфункциях. Данное мера справделива и для рекурсивных функций;

5) *percall* – *cumtime* деленная на количество вызовов;

6) $filename:lineno(function)$ – название функции.

Таблица 3. Список самых затратных функций

ncalls	tottime	percall	cumtime	percall	filename:lineno(function)
504	14.768	0.029	14.768	0.029	{method 'read' of 'cv2.VideoCapture' objects}
451	14.145	0.031	14.145	0.031	{method 'update' of 'cv2.Tracker' objects}
6	11.460	1.910	11.460	1.910	{built-in method _pywrap_tensorflow_internal. TF_SessionRun_wrapper}
504	3.092	0.006	3.092	0.006	{waitKey}
1891	2.538	0.001	6.337	0.003	/usr/lib/python3.5/inspect.py:690(getmodule)
3481286	2.039	0.000	2.040	0.000	{built-in method builtins.hasattr}
135	1.254	0.009	1.254	0.009	{method 'tostring' of 'numpy.ndarray' objects}
3423410	0.815	0.000	1.100	0.000	/usr/lib/python3.5/inspect.py:63(ismodule)
7708	0.618	0.000	0.618	0.000	{method 'ParseFromString' of 'google.protobuf.pyext._message.CMessage' objects}
959	0.597	0.001	0.597	0.001	{imshow}

Как видно из таблицы самыми затратными функциями являются получение изображение с камеры, работа АС и НС. Следовательно в случае

оптимизации, в первую очередь необходимо сконцентрироваться именно на этих частях программах.

3.4.4 Время обработки тестовых видео

Время обработки видео сильно коррелируют с FPS, но в то же время позволяет убедиться, что FPS рассчитан правильно. Данная информация представлена в таблице 4.

Таблица 4. Время обработки тестовых видео

Характеристика	**Куст 1**	**Куст 2**	**Дерево и куст**	**Деревья**	**Поворот и ходьба**
Длительность исходного видео, с	26	23	13	23	30
Время обработки видео, с	124	122	60	92	223
Увеличение времени, раз	4.77	5.30	4.62	4.00	7.43

Все исходные видео имеют частоту 15 кадров в секунду. Посчитанное среднее увеличение в 5.22 раз позволяет заключить что FPS должен быть на уровне 3. Но стоит учитывать, что во время обработки включается время работы НС. FPS рассчитывается по времени, которое прошло между двумя соседними кадрами, поэтому интервал в 3-5 можно считать верным.

3.4.5 Скорость распознавания

Скорость распознавания объектов на видео с помощью НС напрямую влияет на качество сопровождения человеком. Если кадр обрабатывается слишком долго, то робот может потерять объект за время работы. Данные о времени распознавания сведены в таблицу 5. В ячейках таблицы указано время в секундах.

Таблица 5. Скорость распознавания НС объектов на кадре для каждого запуска

Номер прогона	**Куст 1**	**Куст 2**	**Дерево и куст**	**Деревья**	**Поворот и ходьба**
Загрузка	9.29	8.11	10.46	8.71	10.22
1	4.93	2.98	3.01	2.91	2.96
2	4.04	3.45	3.80	2.98	5.13
3	4.25	3.26	4.56	4.40	4.29
4	3.49	3.48	3.83	3.68	3.78
5	3.61	5.02	3.65	3.42	3.59
6	3.58	4.13	3.97	3.49	4.11
7	3.65	3.76	3.65	3.84	3.65
8	2.88	3.63	3.99	3.50	2.84
9	3.68	3.84	3.52	3.22	3.52
10	3.74	2.27	-	3.77	3.47
11	3.52	3.37	-	3.40	3.85
12	3.49	3.66	-	3.61	3.96
13	3.67	3.65	-	-	3.78
14	3.45	3.44	-	-	3.84
15	-	3.72	-	-	3.81
16	-	3.47	-	-	4.08
17	-	3.43	-	-	3.74
18	-	-	-	-	3.38
19	-	-	-	-	3.86
20	-	-	-	-	3.65
21	-	-	-	-	3.50
22	-	-	-	-	3.98
23	-	-	-	-	3.88
Среднее	**3.71**	**3.56**	**3.78**	**3.52**	**3.74**

По итогам оценки было получено что:

1. среднее время распознавания по всем экспериментам 3.66 с;

2. среднее время загрузки НС 9.35 с.

Время распознавания слишком долгое, для работы в реальном времени оно должно находится в пределе 1 с. Среднее время загрузки НС в целом неважно, оно влияет только на первоначальную подготовку сети.

3.4.6 Качество распознавания

Другим важным показателем является качество распознавания людей на кадре. Это можно измерить с помощью нескольких метрик:

1. точность $P = \frac{TP}{TP + FP}$, где TP - истинно положительный результат, FP - ложно положительный результат;

2. полнота $R = \frac{TP}{TP + FN}$, где FN - ложно отрицательный результат;

3. гармоническое среднее из первых двух показателей $F1 = \frac{2 \cdot P \cdot R}{P + R}$.

Под истинно положительным результатом понимается, что НС правильно определила человека. Ложно положительный результат – НС определила человека неверно, ложно отрицательный результат – НС не нашла людей на кадре, хотя они там присутствуют. Точность характеризует способность НС правильно определять человека. Полнота показывает, что НС способна найти всех людей на кадре. Гармоническое среднее представляет среднее между двумя вышеназванными метриками и характеризует работу НС в целом. Данные о точности распознавания представлены в таблице 6.

Таблица 6. Точность распознавания НС для каждого тестового видео

Характеристика	**Куст 1**	**Куст 2**	**Дерево и куст**	**Деревья**	**Поворот и ходьба**
Прогонов НС	13	17	9	12	24
Количество найденных человек	13	12	8	10	24
Количество ненайденных человек	0	5	1	2	0
Ложно положительный результат	2	3	2	2	2
Ложно отрицательный результат	0	3	1	1	0
Точность	0.86	0.80	0.80	0.83	0.92
Полнота	1	0.80	0.88	0.90	1
Гармоническое среднее	0.92	0.8	0.84	0.86	0.96

Данные показатели можно считать удовлетворительными. НС справляется во всех ситуациях.

3.4.7 Качество слежения

Если для НС главным количественным показателем является качество распознавания, то для АС – качество слежения. Для оценки данного показатели были выбранные следующие метрики:

1) время верного слежения за объектом – показывает время, в течение которого алгоритм верного определял центр объекта с учетом возможного некритичного расхождения;

2) количество потерь объекта – сколько раз АС терял объект или неверно определял объект на кадре.

3) доля верного слежения за объектом от общего времен – если первый показатель абсолютный, то этот уже относительный и не связан с длительностью тестового видео.

Все показатели по каждому видео показаны в таблице 7.

Таблица 7. Показатели качества слежения для каждого тестового видео

Характеристика	**Куст 1**	**Куст 2**	**Дерево и куст**	**Деревья**	**Поворот и ходьба**
Время обработки видео, с	124	122	60	92	223
Время верного слежения за объектом, с	64	94	17	22	218
Количество потерь объекта	8	5	3	4	3
Доля верного слежения за объектом от общего времен	0.52	0.77	0.28	0.24	0.98

Из данной таблице видно, что АС не способен хорошо работать в случае долговременного перекрытия человека препятствием, за которым невозможно различить человека. Данный случай требует дополнительной настройки.

3.4.8 Проверка работы алгоритма без НС

Для подтверждения, что внедрение НС в работу АС улучшает качество слежения за человеком, были проведены несколько опытов, в которых не была задействована НС. По итогам опытов была составлена сравнительная таблица 8, в которой в последнем столбце указывается относительное увеличение или уменьшение показателей при использовании НС. Под столбцом с названием АС подразумевается использование только алгоритма слежения и ручное выделение объекта. В столбце с названием АС+НС представлены данные совместной работы алгоритма слежений и нейронной сети.

Таблица 8. Сравнение работы алгоритма при использовании НС

Название характеристики	**АС**	**АС+НС**	**Изменение, %**
Использование процессора, %	168.95	164.35	-2.7
Использование оперативной памяти, МБ	352.25	1259.51	+257.6
Увеличение времени обработки, раз	3.33	5.22	+56.7
Доля верного слежения за объектом от общего времени			
Куст 1	0.40	0.52	+30.0
Куст 2	0.26	0.77	+196.15
Дерево и куст	0.20	0.28	+40.0
Деревья	0.22	0.24	+9
Поворот и ходьба	0.99	0.98	-1

Исходя из данных таблицы можно заключить, что НС требует значительного дополнительного объема оперативной памяти и существенно увеличивает время обработки видео, но позволяет улучшить качество слежения в случае перекрытия человека сильноразреженными объектами, такими как кусты.

3.5 Выявленные проблемы

В ходе тестирования были выявлены несколько наиболее важных проблем, которые были сгруппированы по разделам с демонстрацией их влияния. В левой части изображения представлен результат работы НС, в правой части – результат работы АС. На правой части изображения также отображается траектория центра объекта.

3.5.1 Тряска камеры

Данная проблема критична для АС. Большей частью из-за нее алгоритм не может нормально работать, так как фон должен быть преимущественно статичен. Это связано с тем, что АС ищет объект в определенном области изображения. Результат этой проблемы продемонстрирован на рис. 15.

Рисунок 15 – Проблема тряски камеры

В добавок к этому, НС не способна верно найти человека, в результате чего АС теряет человека.

3.5.2 Малый контраст

Солдат должен сливаться с местностью и иметь как можно меньше отличий от окружающих объектов. Это не позволяет АС однозначно идентифицировать человека, так как деревья и кусты были тоже достаточно похоже на человека в виду их контраста. Поэтому АС часто переключался на них. Это отчетливо видно на рис. 16.

Рисунок 16 – Проблема недостаточного контраста человека

3.6 Идеи для улучшения

Для исправления вышеназванных проблем предлагается несколько идей.

1. Введение экспоненциального фильтра. Причем не только на положение объекта, но и на его скорость. По тестовым видео видно, что человек передвигается большей частью с одинаковой скоростью, без резких скачков. Это позволяет наложить ограничения на возможное положение объекта в следующим кадре.

2. Стабилизация камеры. Это позволит убрать проблему тряски видео и добиться более высоких показателей в вопросе качества слежения.

ЗАДАНИЕ ДЛЯ РАЗДЕЛА
«ФИНАНСОВЫЙ МЕНЕДЖМЕНТ, РЕСУРСОЭФФЕКТИВНОСТЬ И РЕСУРСОСБЕРЕЖЕНИЕ»

Студенту:

Группа	ФИО
8Е41	Волков Артем Алексеевич

Институт	ИШИТР	Кафедра	ОАР
Уровень образования	Бакалавриат	Направление/специальность	15.03.06 Мехатроника и робототехника

Исходные данные к разделу «Финансовый менеджмент, ресурсоэффективность и ресурсосбережение»:	
1. *Стоимость ресурсов научного исследования (НИ): материально-технических, энергетических, финансовых, информационных и человеческих*	*- Тариф на электроэнергию – 5,257 руб./кВт·ч.;* *- Оклад студента – 1850 руб. в месяц;* *- Оклад руководителя проекта – 13824 руб. в месяц.* *- Человеческие ресурсы – 2 человека (руководитель и студент-дипломник).*
2. *Нормы и нормативы расходования ресурсов*	*- Годовая норма амортизации составляет 40 %*
3. *Используемая система налогообложения, ставки налогов, отчислений, дисконтирования и кредитования*	*На основании пункта 1 ст.58 закона №212-ФЗ для учреждений осуществляющих образовательную и научную деятельность вводится пониженная ставка – 27,1%*
Перечень вопросов, подлежащих исследованию, проектированию и разработке:	
1. *Оценка коммерческого потенциала, перспективности и альтернатив проведения НИ с позиции ресурсоэффективности и ресурсосбережения*	*- Методы коммерциализации результатов инженерных решений;* *- SWOT-анализ*
2. *Планирование и формирование бюджета научных исследований*	*- Определение трудоемкости выполнения работ;* *- Расчет материальных затрат НТИ;* *- Основная и дополнительная зарплата исполнителей темы;* *- Отчисления во внебюджетные фонды;* *- Накладные расходы;* *- Проведение анализа безубыточности проекта*
3. *Определение ресурсной (ресурсосберегающей), финансовой, бюджетной, социальной и экономической эффективности исследования*	*- Расчет интегрального показателя финансовой эффективности.*
Перечень графического материала *(с точным указанием обязательных чертежей)***:**	
1. *Оценка конкурентоспособности технических решений* 2. *Матрица SWOT* 3. *Альтернативы проведения НИ* 4. *График проведения и бюджет НИ* 5. *Оценка ресурсной, финансовой и экономической эффективности НИ*	

Дата выдачи задания для раздела по линейному графику	01.03.2018

Задание выдал консультант:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
доцент	Петухов Олег Николаевич	к.э.н.		01.03.2018

Задание принял к исполнению студент:

Группа	ФИО	Подпись	Дата
8Е41	Волков Артем Алексеевич		01.03.2018

4 Финансовый менеджмент, ресурсоэффективность и ресурсосбережение

Целью экономического раздела является проведение детального анализа проекта по критериям конкурентоспособности и ресурсоэффективности, оценка перспективности проекта, определение трудоемкости и графика работ, а также расчёт интегрального показателя ресурсоэффективности.

4.1 Потенциальные потребители

Для того, чтобы определить потенциальных потребителей, необходимо определить целевой рынок и провести его сегментирование. Целевым рынком является рынок военной робототехники. В качестве конечных потребителей выступают военные части.

4.2 Анализ конкурентных решений

Конкурентные решения используют различные датчики для определения лидера. Среди датчиков есть GPS-приемники, лидары, системы технического зрения (СТЗ). При этом для следования за лидером в большинстве случаев используются избыточное количество оборудования. С целью уменьшения количество бортового оборудования в данной работе предлагается новый способ следования. По сравнительному анализу конкурентных решений была составлена таблица 8. Полужирным шрифтом выделен сегмент, на который направлена данная работа.

Таблица 8 – Карта сегментации рынка

		Дополнительное использование СТЗ	
		Да	Нет
Первичный датчик	Лидар	Lockheed Martin SMSS	BigDog
	GPS-приёмник	HDT Global Protector	Northrop Grumman Camel
	СТЗ	-	**Отсутствуют**

4.3 Технология QuaD

Технология QuaD (QUality ADvisor) - это инструмент измерения характеристик, который описывает качество новой разработки, а также ее перспективность на рынке. Технология позволяет принимать решение о целесообразности вложения капитала в НИР и может использоваться при проведении различных маркетинговых исследований, существенным образом снижая их трудоемкость и повышая точность и достоверность результатов. Оценочная карта представлена в таблице 9.

Таблица 9 - Оценочная карта для сравнения конкурентных технических решений

Критерии оценки	Вес критерия	Баллы	Максимальный балл	Относительное значение (3/4)	Средневзвешенное значение (5x2)
1	2	3	4	5	
Показатели оценки качества разработки					
1. Энергоэффективность	0,05	80	100	0,80	0,04
2. Помехоустойчивость	0,04	60	100	0,60	0,024
3. Надежность	0,07	80	100	0,80	0,056
4. Мобильность	0,16	70	100	0,70	0,16
5. Персонализация	0,06	100	100	1,00	0,06
6. Безопасность	0,16	70	100	0,70	0,16

7. Функциональная мощность	0,09	80	100	0,80	0,072
8. Простота эксплуатации	0,09	100	100	1,00	0,09
9. Качество интеллектуального интерфейса	0,08	40	100	0,40	0,034
10. Ремонтопригодность	0,08	50	100	0,50	0,044
Показатели оценки коммерческого потенциала разработки					
11. Перспективность рынка	0.05	100	100	1,00	0,05
12. Цена	0,07	100	100	1,00	0,07
Итого	**1**			**9,9**	**0,823**

Оценка качества и перспективности по технологии QuaD определяется по формуле:

$$П_{ср} = \sum П_i \cdot 100 = 0,823 \cdot 100 = 82,3$$, где:

1) $П_{ср}$ – средневзвешенное значение показателей качества и перспективности научной разработки;

2) $П$ – средневзвешенное значение показателя.

Значение $П_{ср}$ позволяет говорить о перспективах разработки и качестве проведенного исследования. Если значение показателя $П_{ср}$ получилось от 100 до 80, то такая разработка считается перспективной. Если от 79 до 60 – то перспективность выше среднего. Если от 69 до 40 – то перспективность средняя. Если от 39 до 20 – то перспективность ниже среднего. Если 19 и ниже – то перспективность крайне низкая. Из таблицы можно сделать вывод, что разработку можно считать перспективной.

4.4 SWOT – анализ

SWOT – Strengths (сильные стороны), Weaknesses (слабые стороны), Opportunities (возможности) и Threats (угрозы) – это комплексный анализ научно-исследовательского проекта. Такой анализ применяют для исследования внешней и внутренней среды проекта. Составленная матрица SWOT представлена в таблице 10.

Таблица 10 - SWOT-анализ

	Сильные стороны научно-исследовательского проекта:	**Слабые стороны научно-исследовательского проекта:**
	С1. Невысокая стоимость. С2. Мобильность. С3. Надежность. С4. Экономичность.	Сл1. Недостаток конструкторского опыта разработки мобильных роботов. Сл2. Отсутствие финансирования. Сл3. Сложность создания первого прототипа робота. Сл4. Отсутствие репутации на рынке.
Возможности: В1. Использование инновационной инфраструктуры ТПУ. В2. Заинтересованность со стороны государства. В3. Продажа продукта по частям, не только в качестве робота, но и отдельно программное обеспечение.	При использовании инфраструктуры ТПУ можно снизить стоимость проекта. Новизна разработки состоит именно в заложенном алгоритме управления, что позволяет продавать только алгоритм, а не робота целиком.	Появление дополнительного спроса на рынке может способствовать финансированию проекта. Проводя исследования на базе ТПУ можно разработать прототип и получить известность на рынке.
Угрозы: У1. Отсутствие спроса на более производительный алгоритм следования У2. Низкая скорость изготовления. У3. Введение требований к продаваемому ПО. У4.Несвоевременное финансовое обеспечение.	Невысокая цена и хорошая функциональность способствуют повышению спроса. Требования государства могут быть перекрыты заданными ограничениям по надежности.	Необходимо разработать прототип для того, чтобы выйти на рынок и заполучить репутацию.

4.5 Морфологический анализ

Морфологический подход основан на систематическом исследовании всех теоретически возможных вариантов, которые вытекают из закономерностей объекта исследования. Анализ охватывает все возможные варианты. Путем комбинирования вариантов можно получить большое количество различных решений, ряд которых представляет практический интерес. Составим таблицу 11, в которой будут отражены возможные варианты исполнения по различным проблемам разработки.

Таблица 11 - Морфологическая матрица для алгоритма компьютерного зрения

Характеристика	**Варианты исполнения**		
	1	**2**	**3**
Язык программирования	Python	C++	Matlab
Вычислительная платформа	Персональный компьютер	Система на чипе	Удаленный сервер
Камера	Стереокамера	Промышленная камера	Веб камера

Все высокопроизводительные алгоритмы пишутся на статически типизированных языках программирования, таких как C++ и Matlab. Но данные языки не позволяют быстро получить рабочий прототип для тестирования. Робот должен быть самодостаточным, а также в достаточной мере автономным. Камера является единственным источником информации, поэтому информация с неё должна быть максимально информативной. Поэтому в качестве варианта исполнения берётся Python, система на чипе и промышленная камера.

4.6 Планирование научно-исследовательских работ

4.6.1 Структура работ в рамках научного исследования

Для организации НИР применяются различные методы экономического планирования с целью более эффективного использования времени и рабочей силы, а также снижения трудозатрат. Планирование НИР заключается в:

1. составлении перечня работ, необходимых для достижения поставленной задачи;

2. определении участников;

3. установлении продолжительности в рабочих днях;

4. построения линейного графика и его оптимизации.

Примерный порядок этапов и работ, распределение исполнителей по данным видам работ приведен в таблице 12. В таблице и далее используются следующие сокращения:

1. НР – научный руководитель;

2. С – студент.

Таблица 12 – Перечень этапов, работ и распределение исполнителей

Основные этапы	**№ этапа**	**Содержание работ**	**Исполнитель**
Разработка задания	1	Постановка задачи	НР
Выбор направления исследования	2	Обзор научно-технической базы	НР, С
	3	Разработка и утверждение ТЗ	НР, С
	4	Составление календаря проекта	С
	5	Разработка вариантов исполнения проекта	НР, С
	6	Разработка алгоритма работы	С

Теоретические исследования	7	Проектирование архитектуры программы	С
	8	Программирование и отладка алгоритма	С
Экспериментальные исследования	9	Тестирование работы алгоритма	С
	10	Обработка полученных результатов	НР, С
Оформление отчета по НИР	11	Составление пояснительной записки	С
	12	Оформление графического материала	С

На первом этапе происходит постановка цели и задачи исследования – разработка и реализация алгоритмов распознавания изображений по видеопотоку в задачах слежения за объектами. Тематика выбирается научным руководителем и обсуждается со студентом.

На втором этапе студент производит поиск научной литературы по предоставленной тематике для ознакомления и изучения необходимого материала. В дальнейшем данный материал будет использоваться для проведения исследований и разработки устройства.

На третьем этапе студент совместно с научным руководителем разрабатывают общее содержание ВКР. Данный документ является основополагающим при проведении дальнейшего исследования и разработки.

На четвертом этапе составляется календарный план выполнения работ с учётом линейного графика обучения.

На пятом этапе осуществляется разработка архитектуры программного продукта. Для этого составляются требования к функционалу программы. Далее выбирается парадигма программирования. Для упрощения внесения

будущих изменений функционал программы разбивается на несколько независимых друг от друга компонент. Для данных компонент составляются диаграммы классов и блок-схемы функций для визуализации порядка работы алгоритма.

На шестом этапе студентом пишется код для программы на выбранном языке программирования. Затем на этапе тестирования происходит оценка работы алгоритма: быстродействие, точность, робастность.

На восьмом, девятом и десятом этапе студент, под руководством научного руководителя занимается интерпретацией и обработкой результатов, а также оформлением пояснительной записки и графического материала (графические материалы результатов исследования, презентация проекта).

4.6.2 Определение трудоемкости выполнения работ

Наиболее ответственной частью экономических расчетов по теме является расчет трудоемкости работ, так как трудовые затраты составляют основную часть стоимости НИР. Под трудоемкостью работ понимают максимально допустимые затраты труда в человеко-днях на выполнение НИР с учетом организационно технических мероприятий, обеспечивающих наиболее рациональное использование выделенных ресурсов.

Так как отсутствует нормативная база по проводимым работам, а также достоверная информация о процессе выполнения подобных работ иными исполнителями, воспользуемся экспертным способом оценки продолжительности выполнения запланированных работ.

Определим ожидаемое время проведения работ, длительность этапов в рабочих и календарных днях, по формулам, воспользовавшись формулой:

$$t_{ож} = \frac{3 \cdot t_{\min} + 2 \cdot t_{\max}}{5}, \text{ где:}$$

1) $t_{ож}$ – ожидаемое время выполнения i -го этапа работ;

2) $t_{\min}$ - минимально возможная трудоемкость выполнения заданной i -ой работы (оптимистическая оценка: в предположении наиболее благоприятного стечения обстоятельств);

3) $t_{\max}$ - максимально возможная трудоемкость выполнения заданной i-ой работы (пессимистическая оценка: в предположении наиболее неблагоприятного стечения обстоятельств).

Ожидаемое, минимальное и максимальное время исполнения в предложенной выше формуле, оцениваются в рабочих днях на человека. Произведем перевод этих величин в календарные дни, воспользовавшись следующей формулой:

$T_{КД} = T_{РД} \cdot T_{К}$, где:

1) $T_{КД}$ – продолжительность выполнения этапа в календарных днях;

2) $T_{К}$ – коэффициент календарности, позволяющий перейти от длительности работ в рабочих днях к их аналогам в календарных днях. Рассчитывается по формуле:

$$T_{К} = \frac{T_{КЛ}}{T_{КЛ} - T_{ВД} - T_{ПД}}, \text{ где:}$$

1) $T_{КЛ}$ – календарные дни ($T_{КЛ} = 365$);

2) $T_{ВД}$ – выходные дни ($T_{ВД} = 52$);

3) $T_{ПД}$ – праздничные дни ($T_{ПД} = 10$).

В свою очередь рабочие дни рассчитываются по следующей формуле:

$$T_{РД} = \frac{t_{ожс}}{К_{ВН}} \cdot К_{Д}, \text{ где:}$$

1) $К_{ВН}$ – коэффициент выполнения работ, учитывающий влияние внешних факторов на соблюдение предварительно определенных длительностей;

$K_Д$ – коэффициент, учитывающий дополнительное время на компенсацию непредвиденных задержек и согласование работ ($K_Д = 1 - 1.2$; в этих границах конкретное значение принимает сам исполнитель).

Для простоты расчетов примем $K_Д$ и $K_{ВН}$, равными единице. Тогда формула для расчета календарных дней преобразуется в следующую:

$$T_{КД} = T_{РД} \cdot T_К = t_{ожс} \cdot T_К = \frac{3 \cdot t_{\min} + 2 \cdot t_{\max}}{5}$$

Воспользовавшись данными из таблицы 4, приведенными выше формулами, произведем расчет продолжительности выполнения работ студентом в календарных днях. Результаты расчетов представлены в таблице 13.

Таблица 13 - Временные показатели проведения научного исследования

Название работы	Трудоемкость работ, чел-дни						Длительность работ в рабочих днях $T_{РД}$		Длительность работ в календарных днях $T_{КД}$	
	$t_{\min}$		$t_{\max}$		$t_{ожс}$		Одновременное выполнение работ		Одновременное выполнение работ	
	Студент	Научный руководитель	Студент	Научный руководитель	Студент	Научный руководитель	Студент	Научный	Студент	Научный
Постановка задачи	5	3	8	6	6,2	4,2	3,1	2,1	5	1
Обзор научно-технической базы	7	2	10	4	12	2,8	4,5	1,4	5	3

Разработка и утверждение ТЗ	7	1	12	2	9	1,2	4,5	0,7	7	2
Составление календаря проекта	3	0	5	0	3,8	0	3,8	0	6	0
Разработка вариантов исполнения проекта	9	2	16	7	11,8	3,2	5,9	2,6	8	4
Разработка алгоритма работы	3	0	8	0	5	0	5	0	8	0
Проектирование архитектуры программы	7	0	14	0	9,8	0	9,8	0	15	0
Программирование и отладка алгоритма	7	0	10	0	8,2	0	8,2	0	13	0
Тестирование работы алгоритма	3	0	7	0	4,6	0	4,6	0	7	0
Обработка полученных результатов	6	4	14	8	9,2	5,6	4,6	2,8	7	5

Составление пояснительной записки	5	0	10	0	7	0	7	0	11	0
Оформление графического материала	10	5	14	8	11,6	6,2	5,8	3,1	9	5
Итого									114	34

4.6.3 Разработка графика проведения исследования

По данным из таблицы 13 создадим диаграмму Ганта.

Таблица 14 - Календарный план-график проведения НИОКР

№	Этап	Исполнитель	$T_{КД}$	Продолжительность выполнения работ																	
				Февраль			Март			Апрель			Май			Июнь					
1	Постановка задачи	НР	1																		
		С	5																		
2	Обзор научно-технической базы	НР	3																		
		С	7																		
3	Разработка и утверждение ТЗ	НР	2																		
		С	7																		
4	Составление календаря проекта	С	6																		
5	Разработка вариантов исполнения проекта	НР	4																		
		С	8																		
6	Разработка алгоритма работы	С	8																		
7	Проектирование архитектуры программы	С	15																		
8	Программирование и	С	13																		

	отладка алгоритма																	
9	Тестирование работы алгоритма	С	7															
10	Обработка полученных результатов	НР	5															
		С	7															
11	Составление пояснительной записки	С	11															
12	Оформление графического материала	НР	5															
		С	9															

4.7 Бюджет научно-технического исследования

Бюджет научно-технического исследования должен быть основан на достоверном отображении всех видов расходов, связанных выполнением проекта. В процессе формирования бюджета разработки используется следующая группировка затрат по статьям:

1. материальные затраты разработки;
2. основная заработная плата исполнителей темы;
3. дополнительная заработная плата исполнителей темы;
4. отчисления во внебюджетные фонды (страховые отчисления);
5. затраты на научные и производственные командировки;
6. накладные расходы.

4.7.1 Расчет материальных затрат

Для вычисления материальных затрат воспользуемся следующей формулой:

$$З_M = (1 + k_t) \cdot \sum_{i=1}^{m} Ц_i \cdot N_{расх.}$$, где:

1) m – количество видов материальных ресурсов;

2) $Ц_i$ – цена приобретения единицы i-го вида потребляемых материальных ресурсов;

3) $N_{расх.}$ – количество материальных ресурсов i-го вида, планируемых к использованию при выполнении научного исследования;

4) k_t – коэффициент, учитывающий транспортно-заготовительные расходы.

Таблица 15 - Материальные затраты

Наименование	**Единица измерения**	**Количество**	**Цена, руб.**
ПК	шт.	1	25000
Лист А4	шт.	100	200
Ручка	шт.	3	90
Камера	шт.	1	5000
Система на чипе	шт.	1	20000
Итого, руб.	**50290**		

4.7.2 Основная заработная плата

Статья включает основную заработную плату работников, непосредственно занятых выполнением НТИ, (включая премии, доплаты) и дополнительную заработную плату. Она рассчитывается по формуле

$$З_{ЗП} = З_{осн.} + З_{доп.}, \text{ где:}$$

1) $З_{осн.}$ – основная заработная плата;

2) $З_{доп.}$ – дополнительная заработная плата (12-20 % от $З_{осн.}$).

Среднедневная заработная плата рассчитывается по формуле:

$$З_{дн.} = \frac{З_M \cdot M}{F_Д}, \text{ где:}$$

1) $З_M$ – месячный должностной оклад работника, руб.;

2) M – количество месяцев работы без отпуска в течение года;

3) $F_Д$ – действительный годовой фонд рабочего времени научно-технического персонала, раб. дн.

Таблица 16 - Баланс рабочего времени

Показатели рабочего времени	**Руководитель**	**Студент**
Календарное число дней	365	365
Количесво нерабочих дней - выходные дни - праздничные дни	120	120
Потери рабочего времени - отпуск - невыходы по болезни	48	72
Действительный годовой фонд рабочего времени	197	173

Месячный оклад работника рассчитывается по формуле:

$$З_М = З_{ТС} \cdot (1 + k_{пр} + k_д) \cdot k_р$$, где:

1) $З_{ТС}$ – заработная плата по тарифной ставке, руб.;

2) $k_{пр}$ – премиальный коэффициент, равный 0,3 (т.е. 30% от Зтс);

3) $k_д$ – коэффициент доплат и надбавок составляет примерно 0,2 – 0,5 (в НИИ и на промышленных предприятиях – за расширение сфер обслуживания, за профессиональное мастерство, за вредные условия: 15-20% от $З_{ТС}$);

4) $k_р$ – районный коэффициент, равный 1,3 (для Томска).

Расчёт основной заработной платы приведён в таблице 17.

Таблица 17 - Расчет основной заработной платы

Исполнители	Разряд	$З_{ТС}$, руб.	$k_{пр}$	$k_д$	$k_р$	$З_М$, руб	$З_{дн.}$, руб.	$Т_Р$, раб. дн.	$З_{осн.}$, руб.
Руководитель	1	9489	0,3	0,2	1,3	18503,55	976,83	24	19862,52
Студент		1854	0	0	1,3	2410,2	130,95	137	13346,43
								Итого $З_{осн.}$	**33208,95**

4.7.3 Дополнительная заработная плата

Дополнительная заработная плата включает заработную плату за не отработанное рабочее время, но гарантированную действующим законодательством. Расчет дополнительной заработной платы ведется по формуле:

$$З_{доп.} = k_{доп.} \cdot З_{осн.}, \text{ где:}$$

1) $k_{доп.}$ – коэффициент дополнительной заработной платы (на стадии проектирования принимается равным 0,12 – 0,15).

$k_{доп.}$ равен 0,12. Результаты по расчетам дополнительной заработной платы сведены в таблицу 18.

Таблица 18 - Затраты на дополнительную заработную плату

Исполнители	Основная зарплата (руб.)	$k_{доп.}$	Дополнительная зарплата (руб.)
Руководитель	19862,52	0,12	2383,50
Студент	13346,43	0,12	1601,57
		Итого	**3985,07**

4.7.4 Отчисления во внебюджетные фонды

Величина отчислений во внебюджетные фонды определяется исходя из формулы:

$$З_{внеб.} = k_{внеб.} \cdot (З_{осн.} + З_{доп.})$$

где kвнеб – коэффициент отчислений на уплату во внебюджетные фонды (пенсионный фонд, фонд обязательного медицинского страхования и пр.).

На 2018 г. в соответствии с Федеральным законом от 24.07.2009 №212-ФЗ установлен размер страховых взносов равный 30%. На основании пункта 1 ст.58 закона №212-ФЗ для учреждений, осуществляющих образовательную и научную деятельность в 2018 году пониженная ставка – 27,1%. Отчисления во внебюджетные фонды представлены в таблице 19.

Таблица 19 - Отчисления во внебюджетные фонды

Исполнитель	Основная заработная плата, руб.	Дополнительная заработная плата, руб.
Руководитель проекта	19862,52	2383,50
Студент	13346,43	1601,57
Коэффициент отчислений во внебюджетные фонды	27,1%	
Итого		
Руководитель	5382,74	
Студент	3616,88	
Итого	**8998,88**	

4.7.5 Накладные расходы

Накладные расходы учитывают прочие затраты организации, не попавшие в предыдущие статьи расходов: печать и ксерокопирование материалов, оплата услуг связи, электроэнергии и т.д. Расчет накладных расходов определяется по формуле:

$$З_{нак.} = \sum Cm \cdot k_{нр}$$, где:

1) $k_{нр}$ – коэффициент, учитывающий накладные расходы;

2) Cm – затраты по статьям накладных расходов.

Величину коэффициента накладных расходов можно взять в размере 10%.

$$З_{нак.} = (50290 + 33208,95 + 3985,07 + 8998,88) \cdot 0,10 = 9647,29 \ руб.$$

4.7.6 Контрагентные расходы

Контрагентные расходы включают затраты, связанные с выполнением каких-либо работ по теме сторонними организациями (контрагентами, субподрядчиками). В данном проекте отсутствует необходимость в стороннем подрядчике.

4.7.7 Формирование бюджета проекта

Рассчитанная величина затрат научно-исследовательской работы является основой для формирования бюджета затрат проекта. Определение

бюджета затрат на научно-исследовательский проект по каждому варианту исполнения приведен в таблице 20.

Таблица 20 – Расчет бюджета затрат НТИ

Наименование статьи	**Сумма, руб.**
1.Материальные затраты НТИ	50290,00
2.Затраты на заработную плату научному руководителю	22246,02
3.Затраты на заработную плату студенту	14948,00
4.Затраты на отчисления во внебюджетный фонд	8998,88
5.Накладные расходы	9647,29
Бюджет затрат НТИ	**106130,19**

4.7.8 Определение эффективности использования ресурсов

В результате исследования были определены затраты на проект по разработке алгоритма компьютерного зрения. Бюджет составляет 106 тыс. руб. Учитывая все конкурентные преимущества данного программного обеспечения, можно предположить, что продукт будет конкурентоспособным и будет иметь спрос на рынке.

Заключение

В ходе данной работы был разработан алгоритм слежения за объектами одного класса - человека. Данный алгоритм был апробирован на тестовых видео, которые были записаны с учетом специфики реального применения. По итогам тестирования были собраны метрики производительности и качества слежения, которые количественно и качественно характеризуют работу алгоритма.

Добавление сверточной нейронной сети к имеющемуся алгоритму слежения позволило решить проблему потери объекта в случае его исчезновения из кадра, а также перекрытии другими объектами или препятствиями. В дальнейшем планируется увеличение производительности и качества работы алгоритма.

Полученные количественные характеристики работы программы показывают невозможность применения данной работы в текущей реализации в реальном проекте, так как быстродействие не находится на достаточном уровне. В добавок к этому, АС не способен работать в быстроизменяющейся среде без добавления дополнительных фильтров. Одним из предложенных решений является установка экспоненциального фильтра на положение и скорость человека.

Приложение А. Дерево вызова процедур

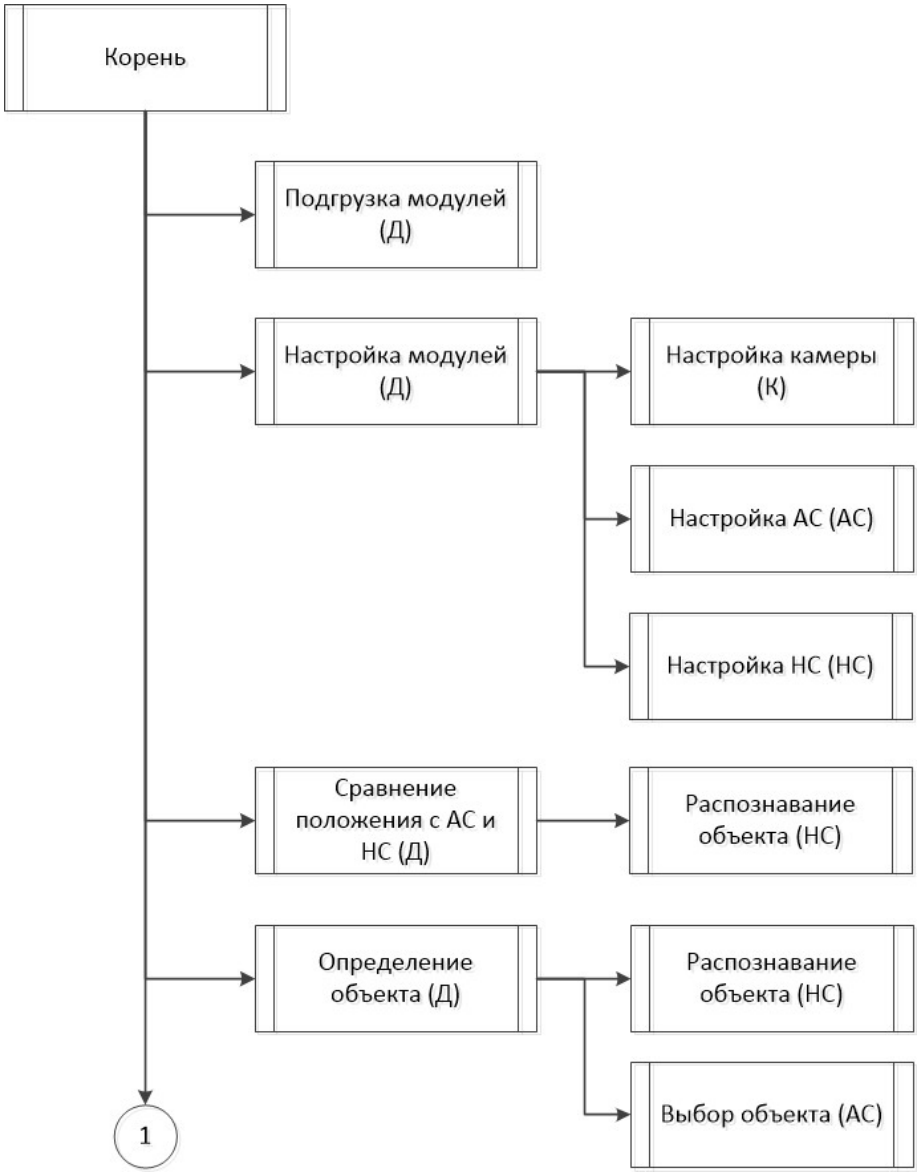

Рисунок 17 – Дерево вызова процедур, часть 1

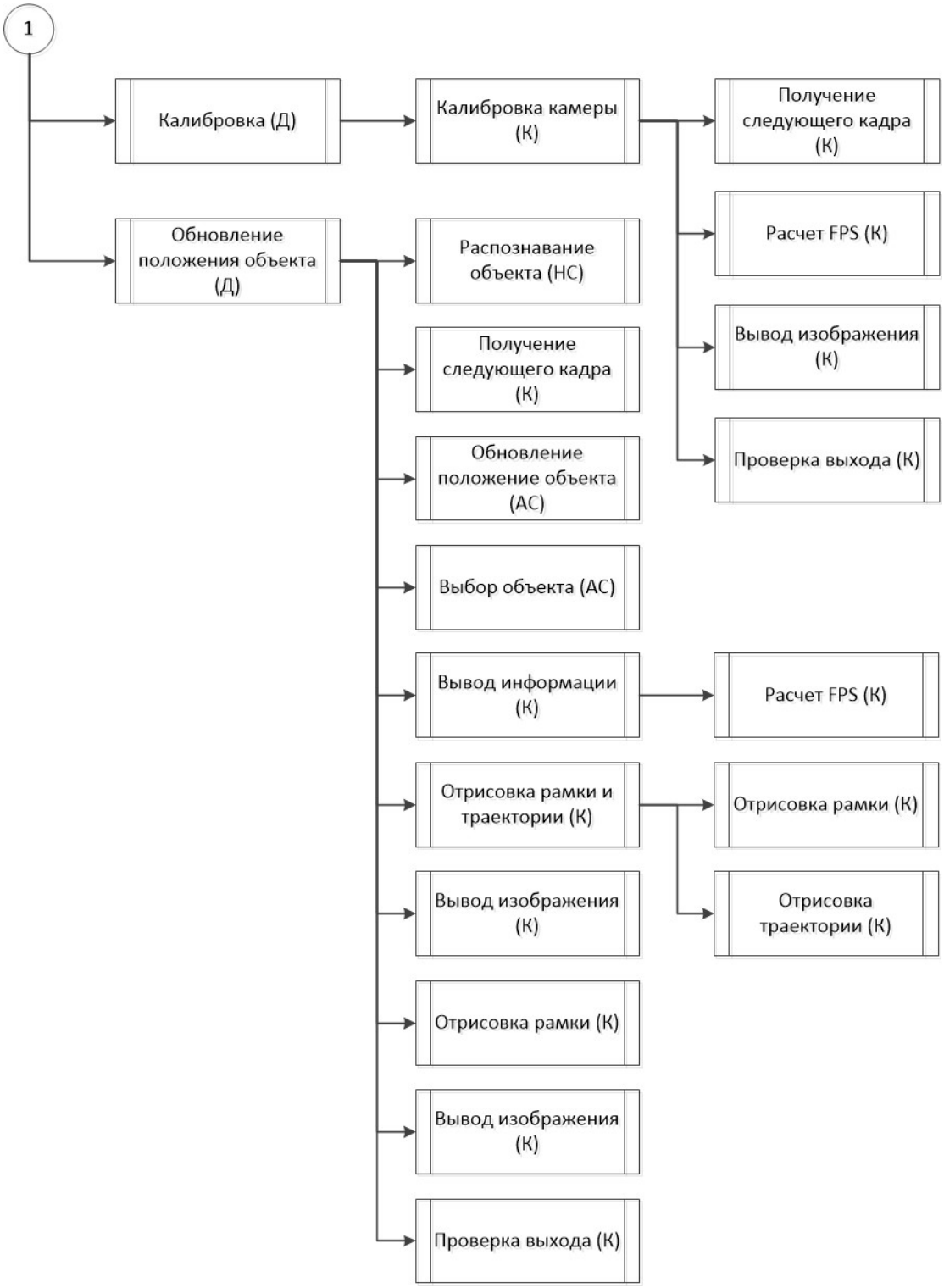

Рисунок 18 – Дерево вызова процедур, часть 2

Приложение Б. Блок-схема главной программы

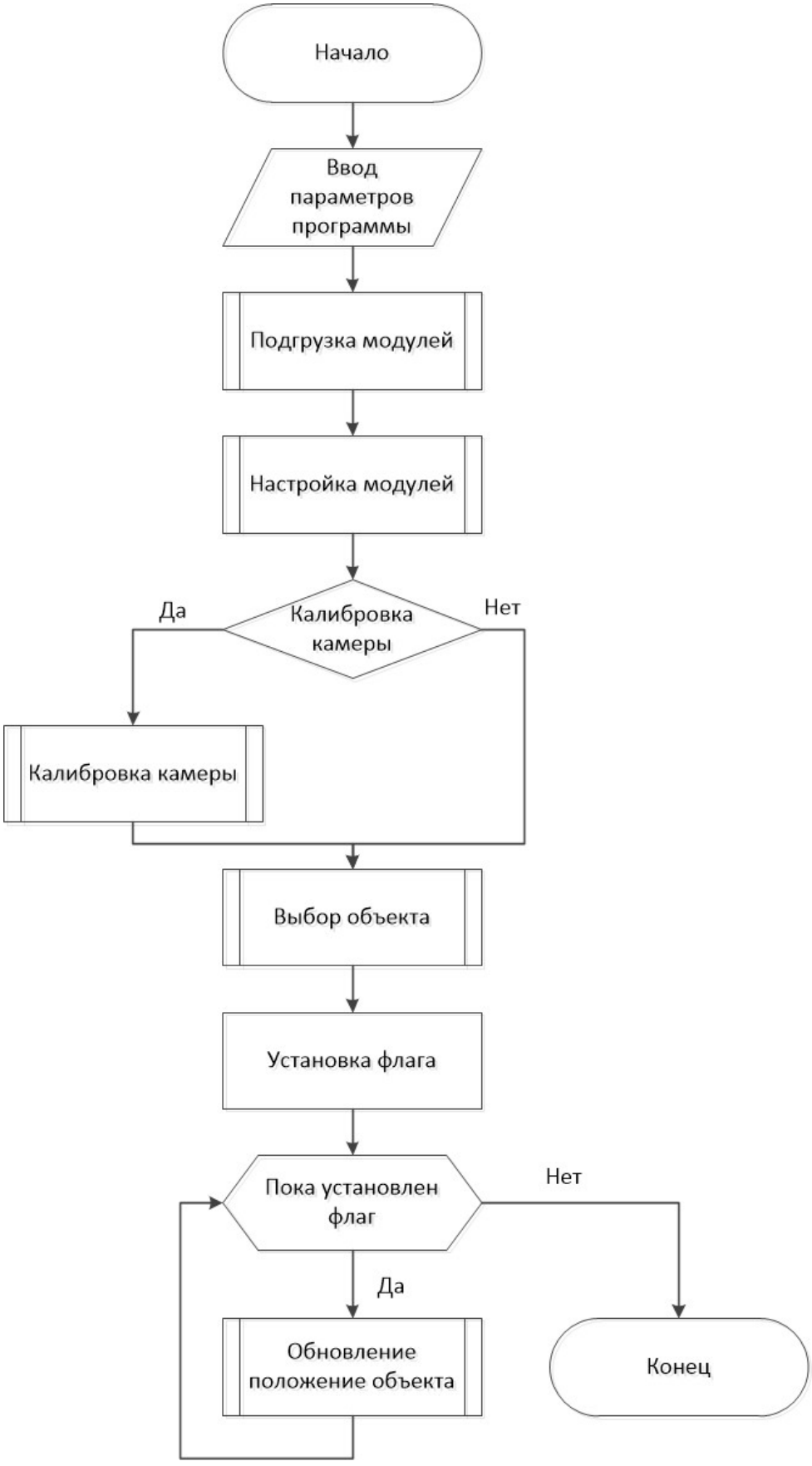

Рисунок 19 - Блок-схема главной программы

Приложение В. Блок-схемы функций модуля Камеры

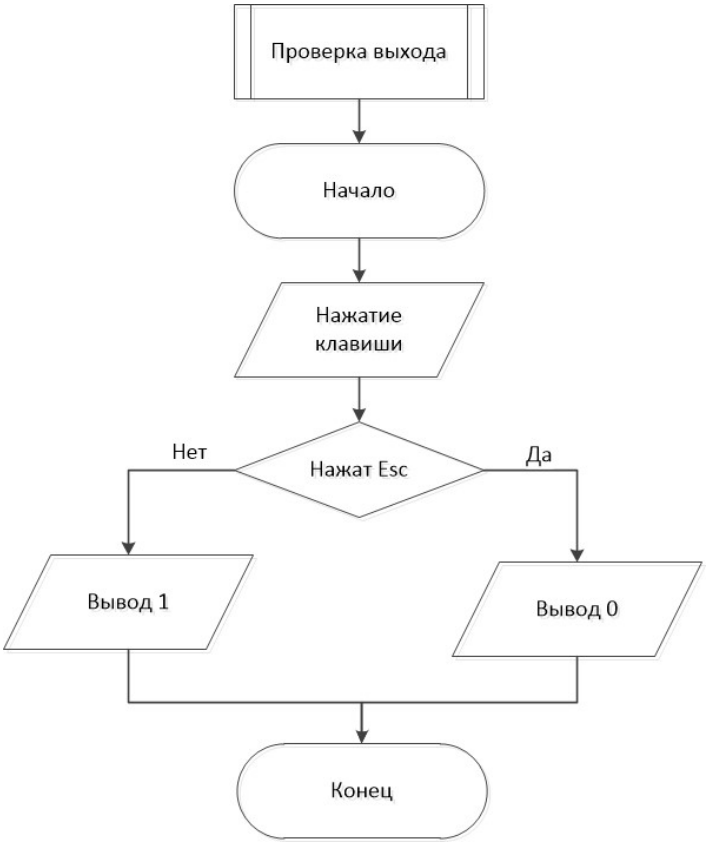

Рисунок 20 – Блок-схема функции проверка выхода

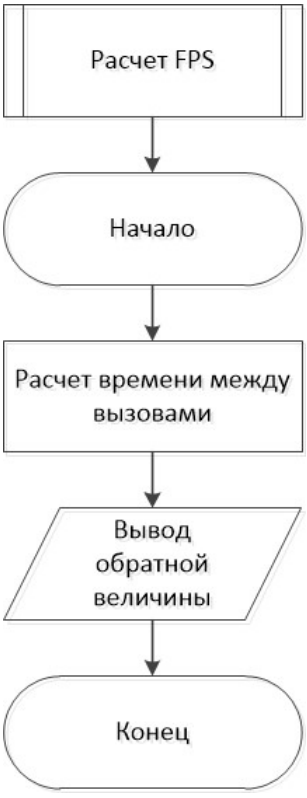

Рисунок 21 – Блок-схема функции расчет FPS

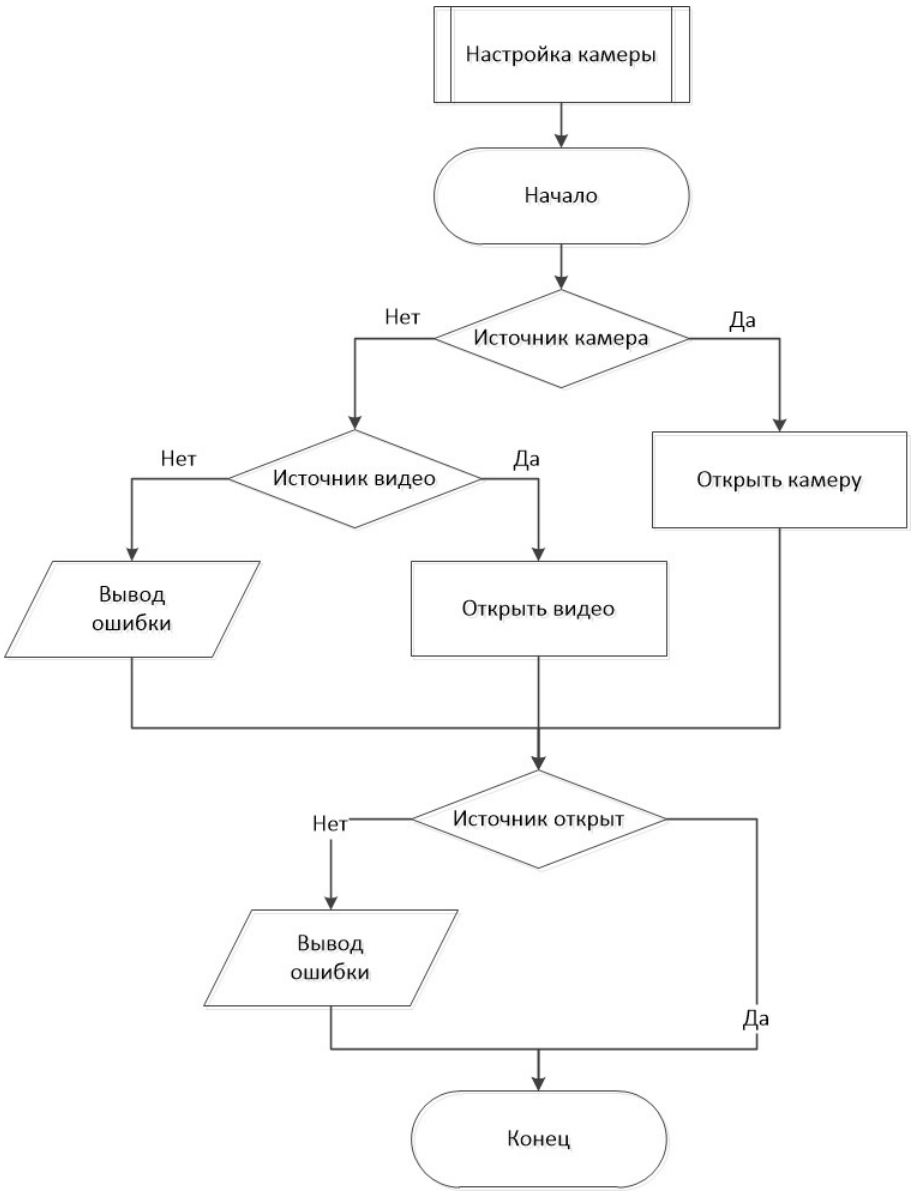

Рисунок 22 – Блок-схема функции настройка камеры

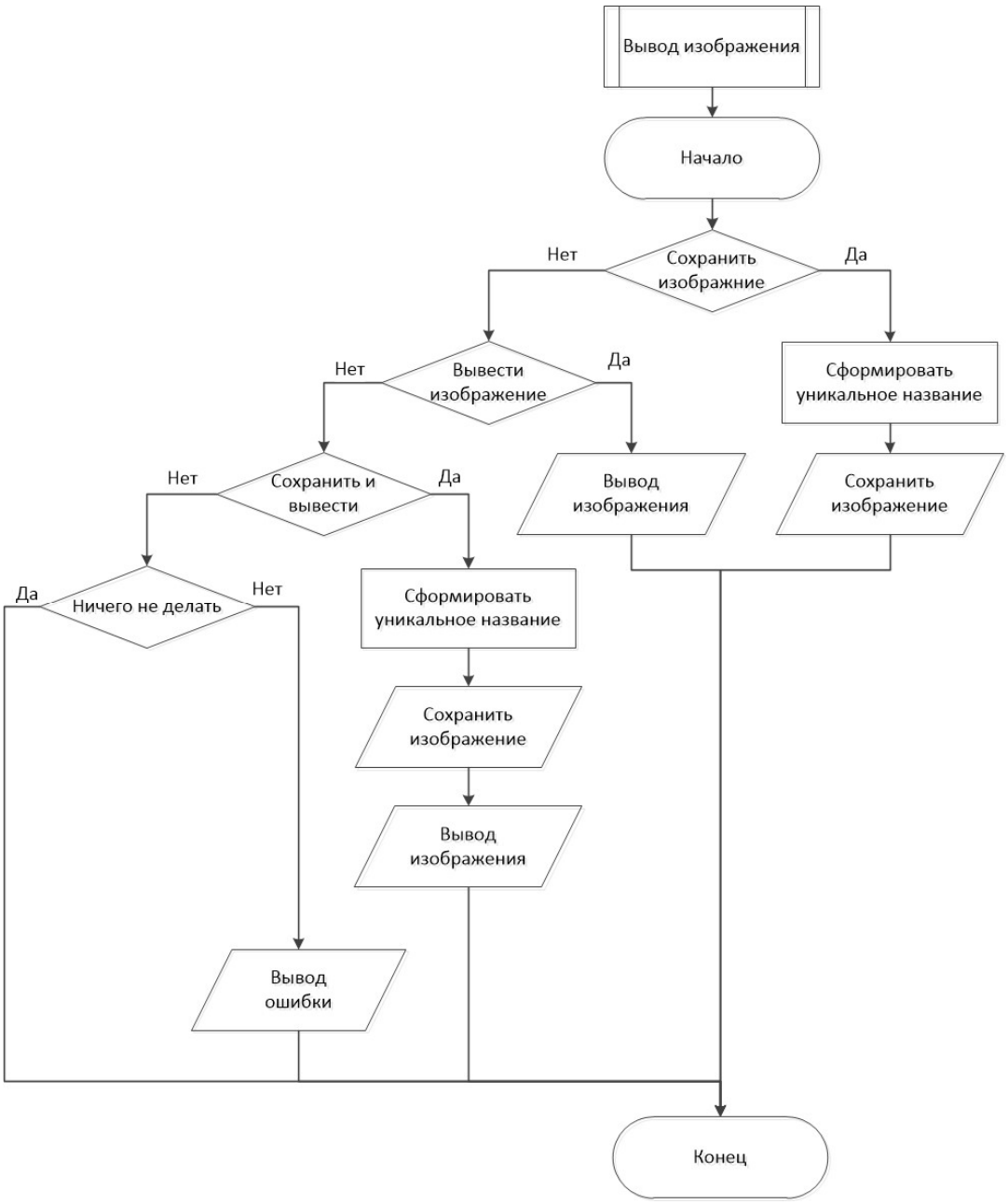

Рисунок 23 - Блок-схема функции вывода изображения

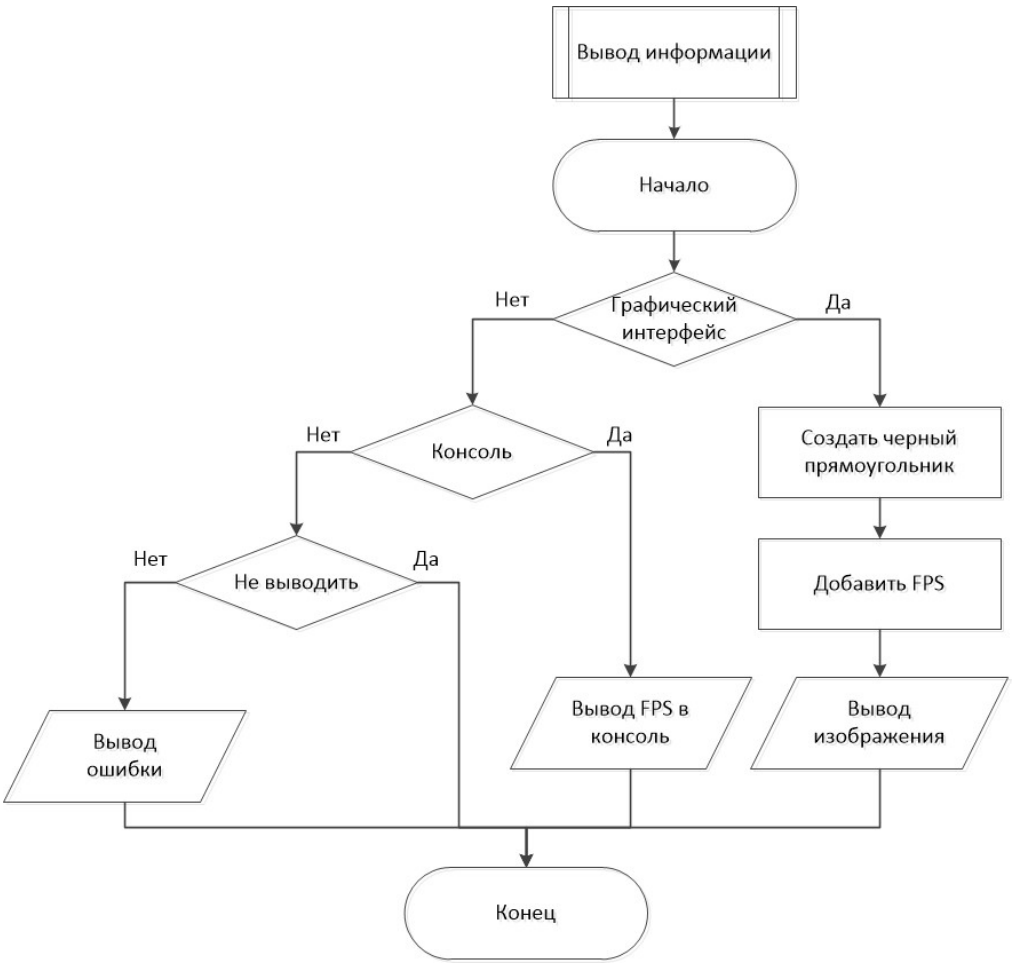

Рисунок 24 - Блок-схема функции вывода информации

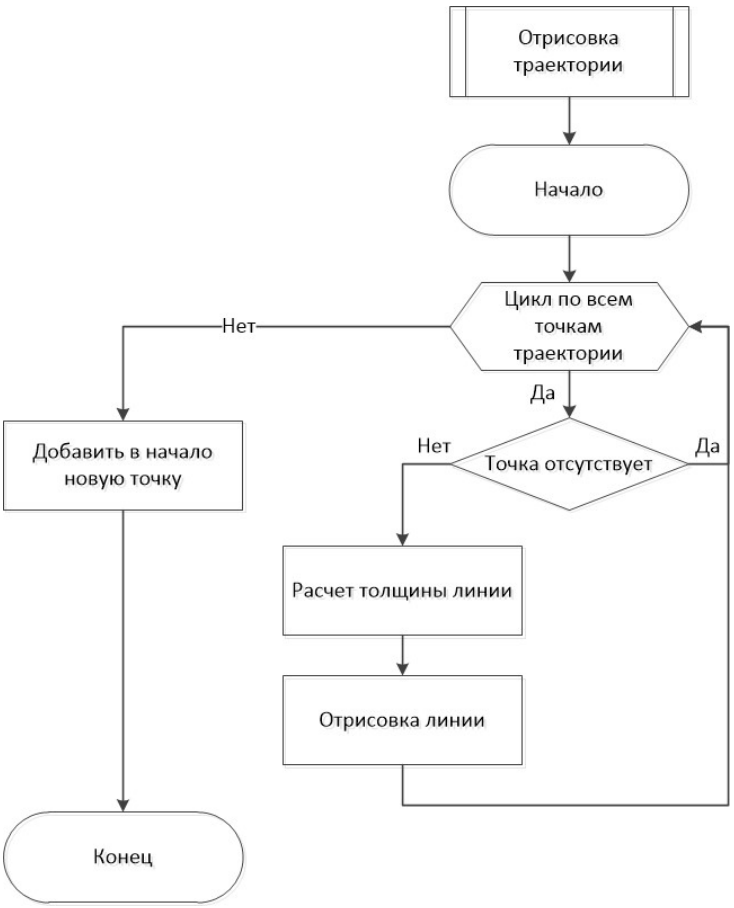

Рисунок 25 - Блок-схема функции отрисовки траектории

Рисунок 26 - Блок-схема функции отрисовки рамки и траектории

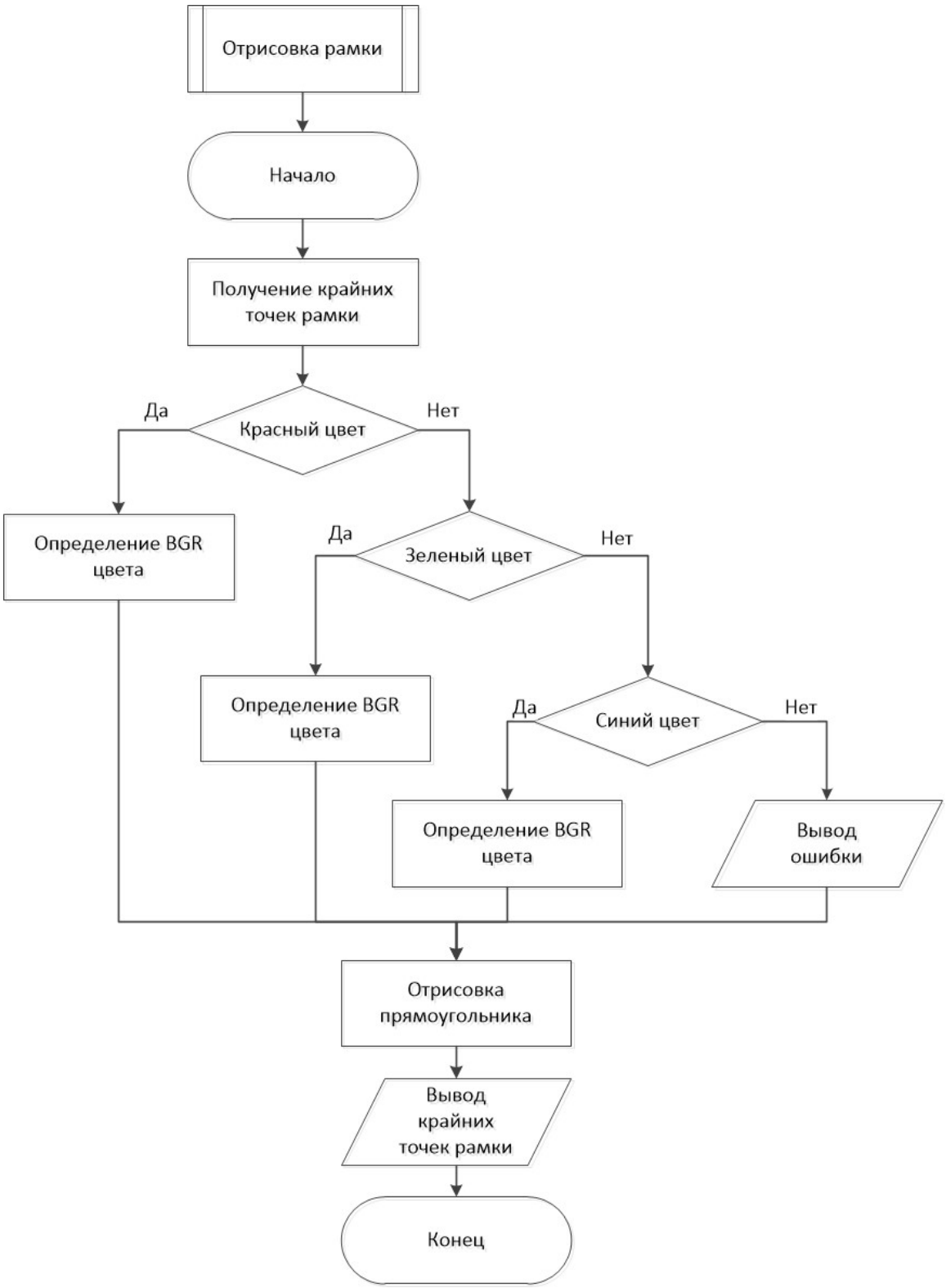

Рисунок 27 - Блок-схема функции отрисовки рамки

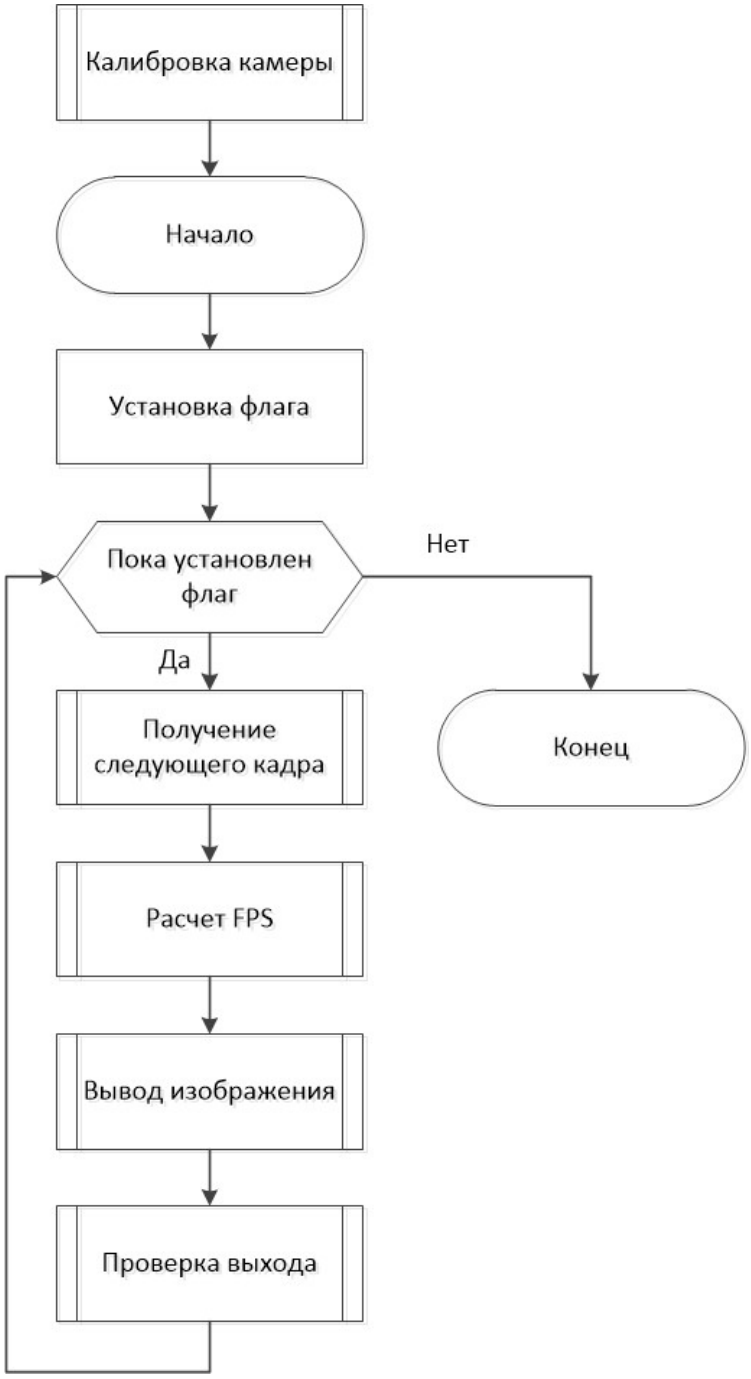

Рисунок 28 - Блок-схема функции калибровки камеры

Приложение Г. Блок-схемы функций модуля НС

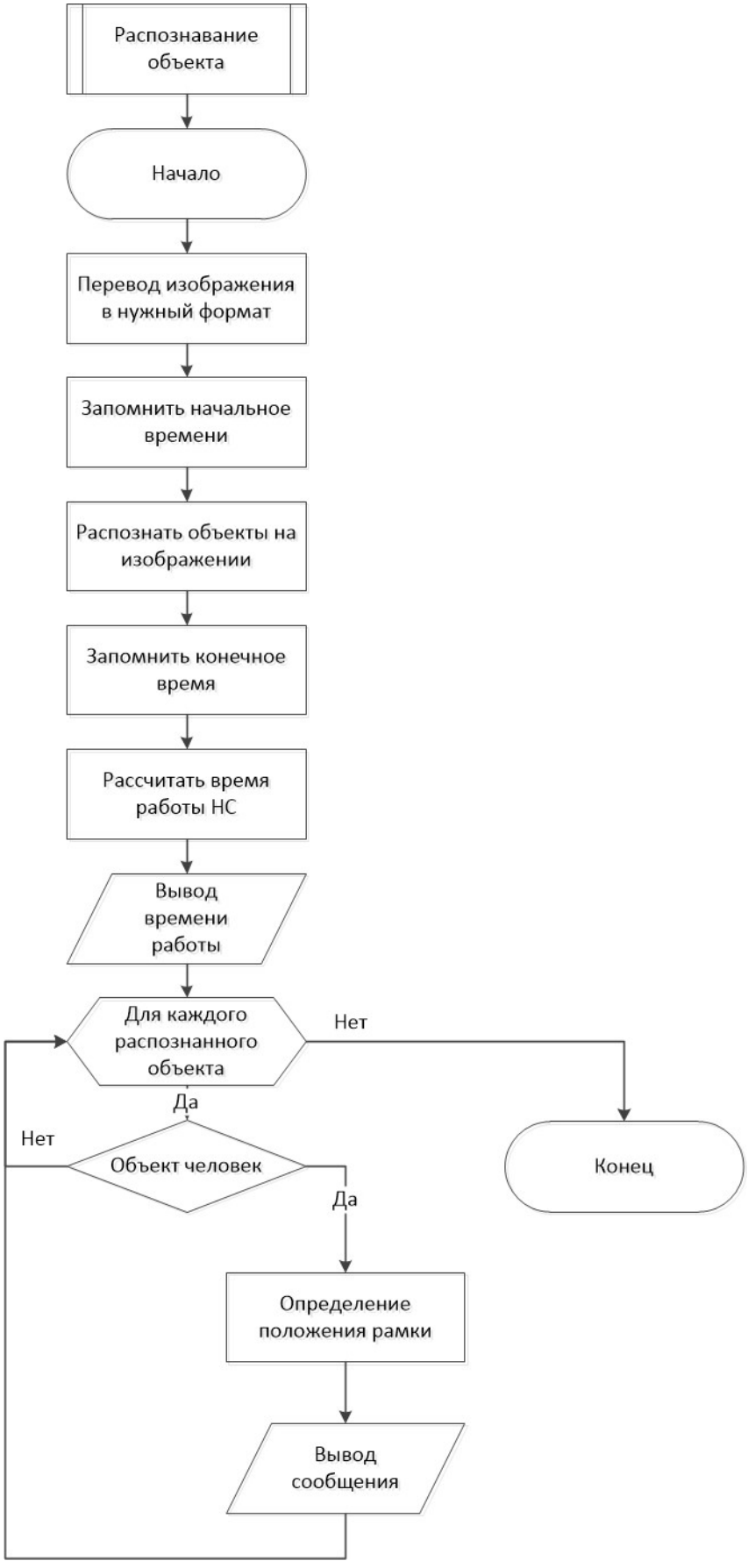

Рисунок 29 – Блок-схема функции распознавания объекта

Рисунок 30 – Блок-схема функции настройка НС

Приложение Д. Блок-схемы функции модуля АС

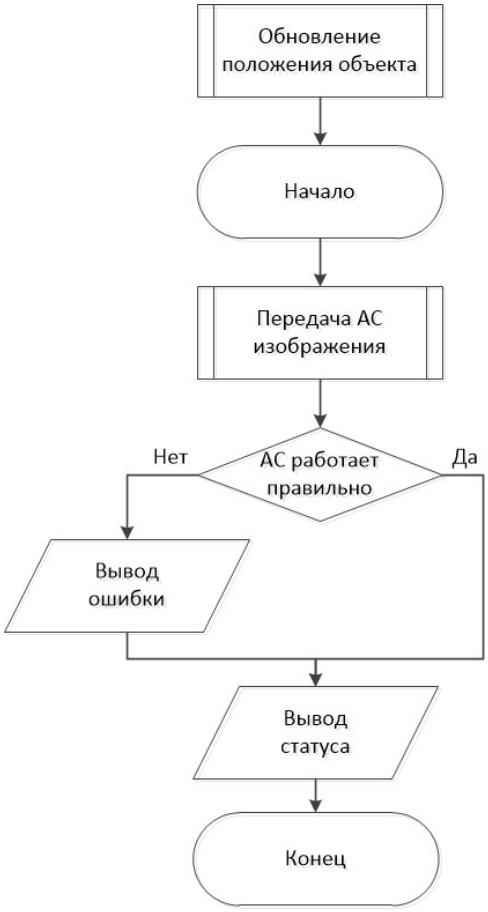

Рисунок 31 – Блок-схема функции обновления положения объекта

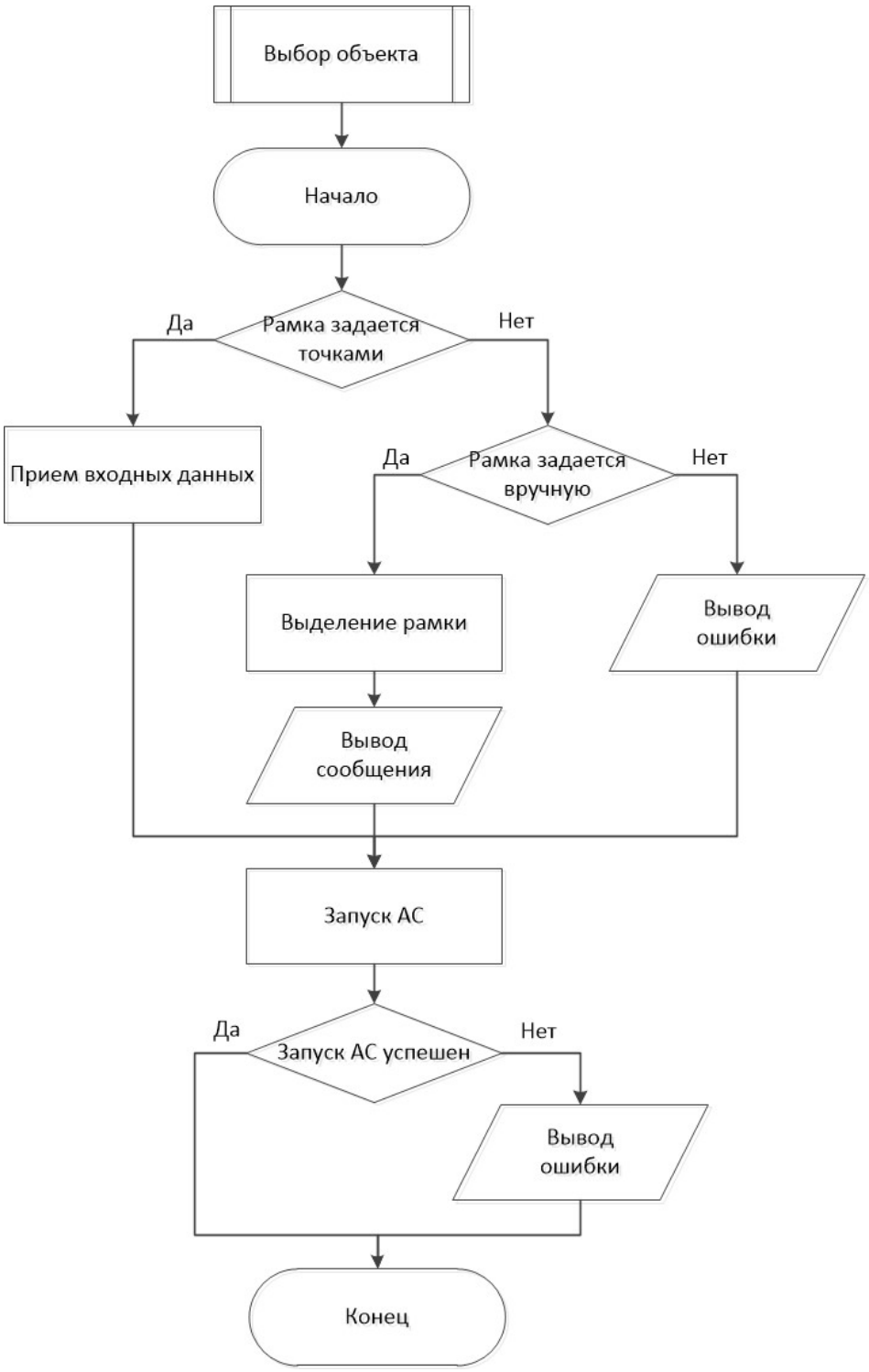

Рисунок 32 – Блок-схема функции выбора объекта

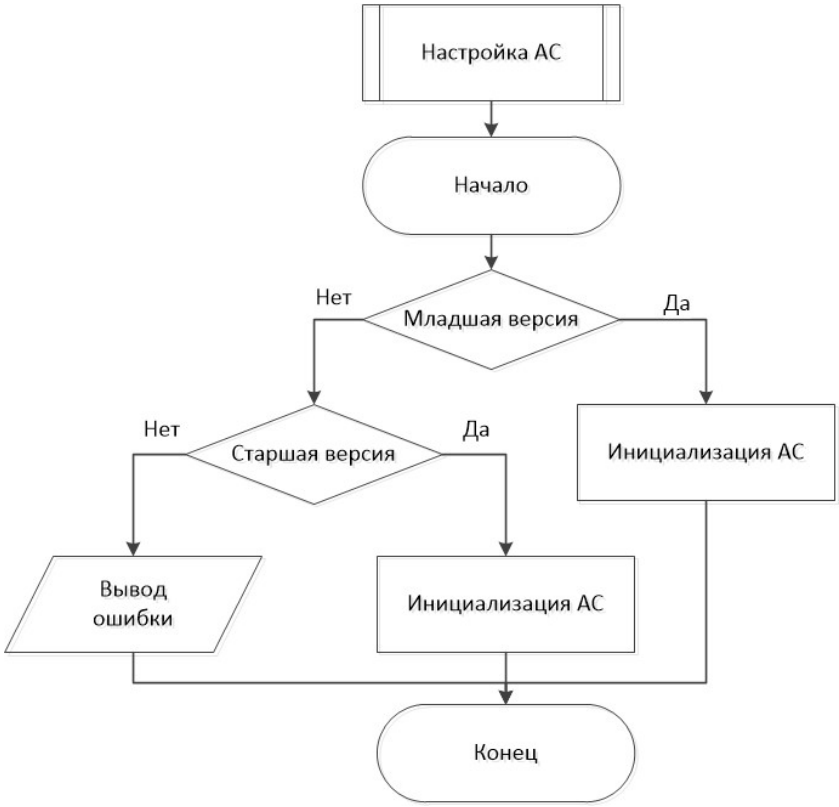

Рисунок 33 – Блок-схема функции настройки АС

Приложение Е. Блок-схемы функции модуля Диспетчера

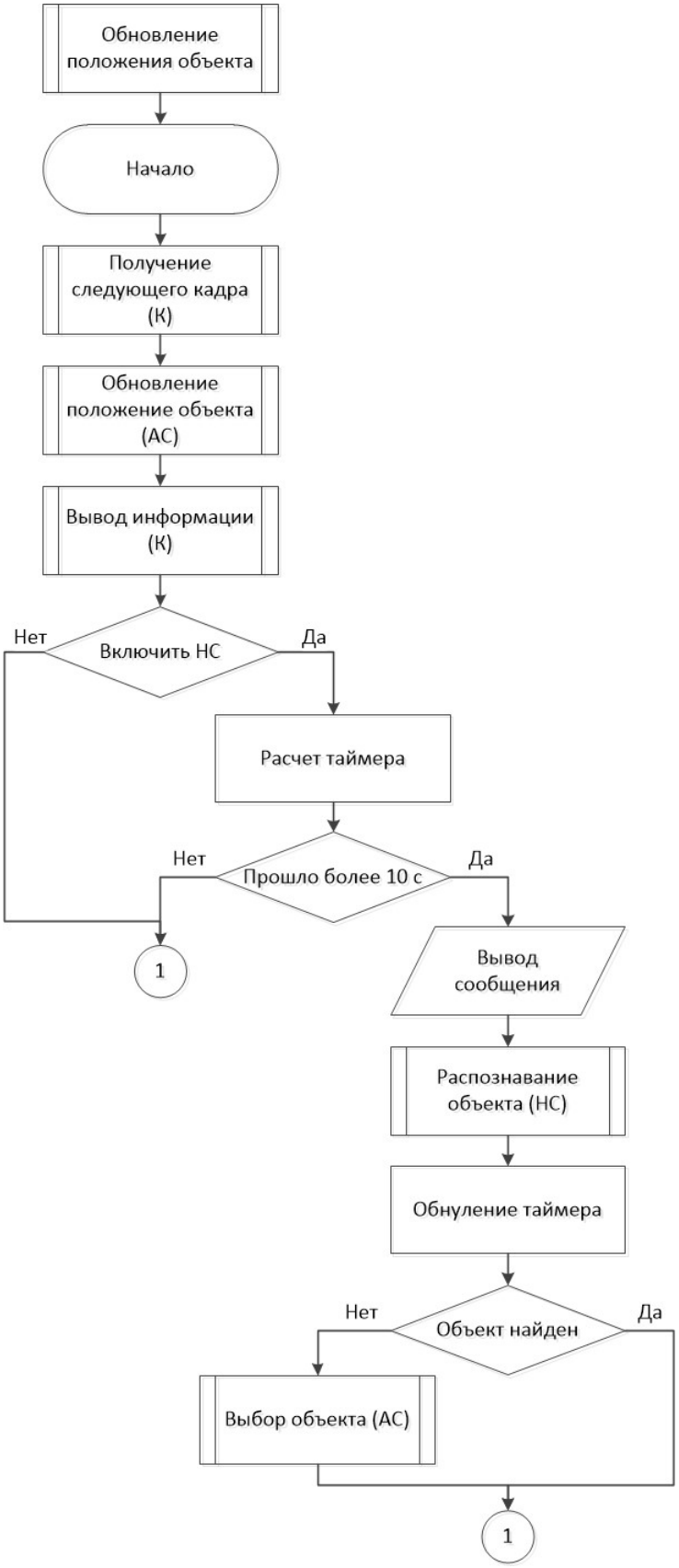

Рисунок 34 – Блок-схема функции обновления положения объекта, часть 1

Рисунок 35 – Блок-схема функции обновления положения объекта, часть 2

Рисунок 36 – Блок-схема функции калибровки

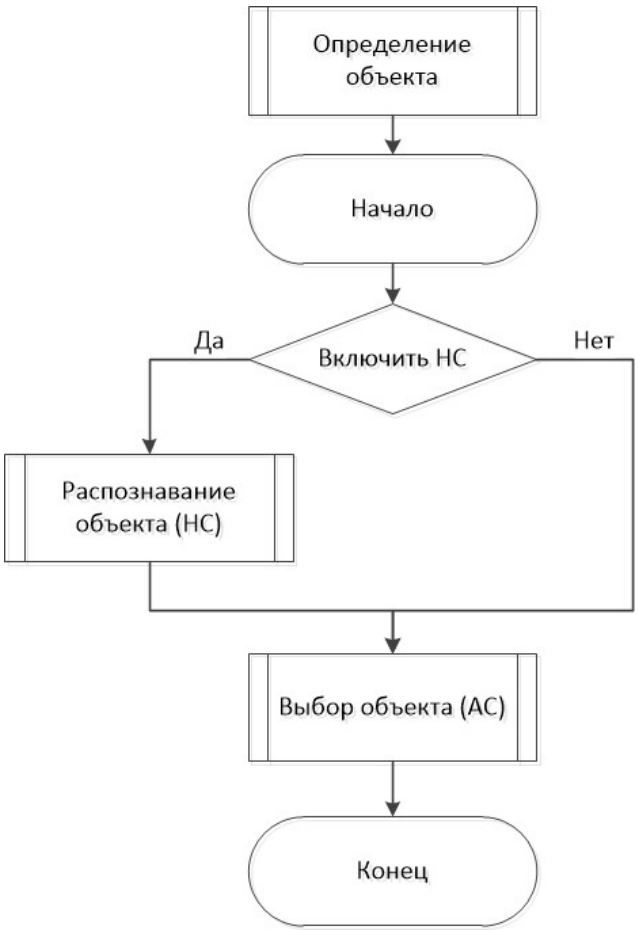

Рисунок 37 – Блок-схема функции определения объекта

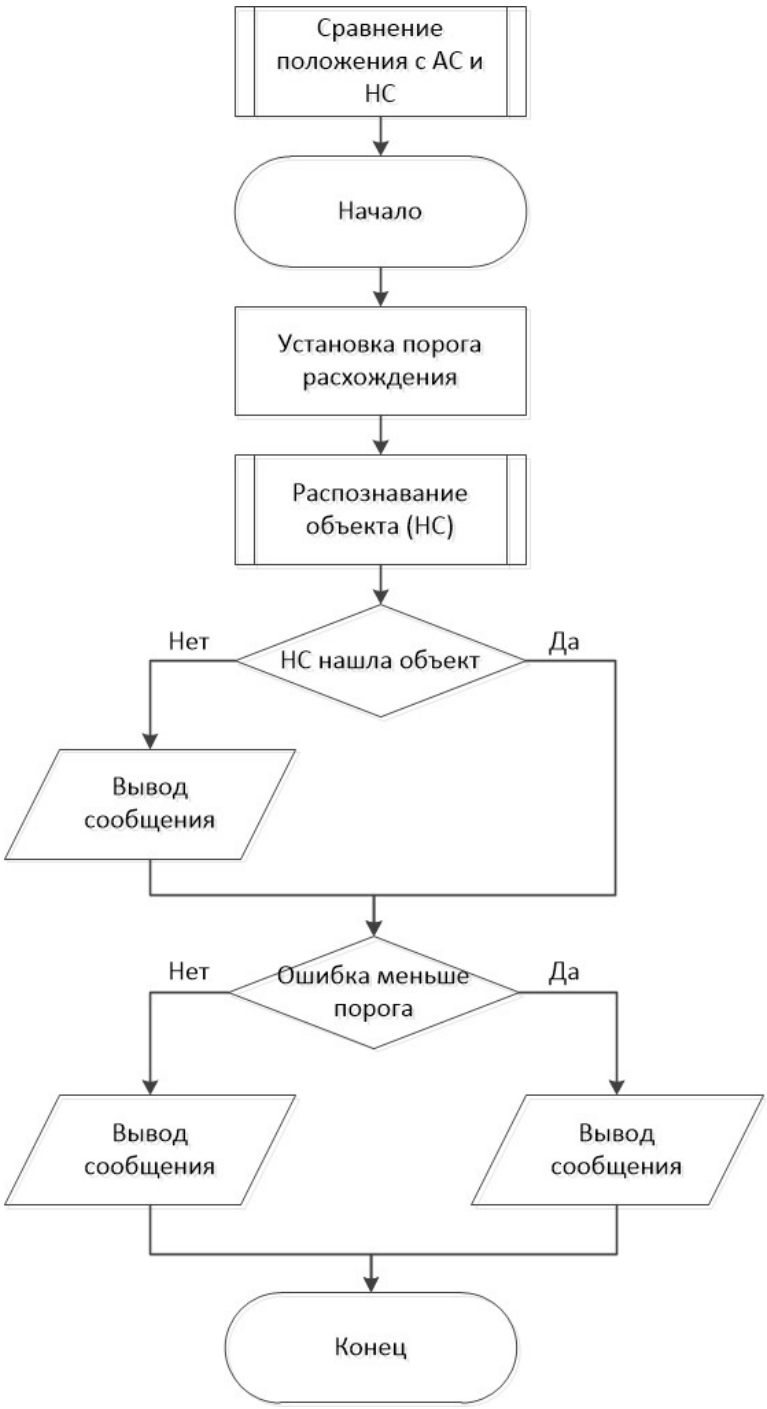

Рисунок 38 – Блок-схема функции сравнения положения объекта с АС и НС

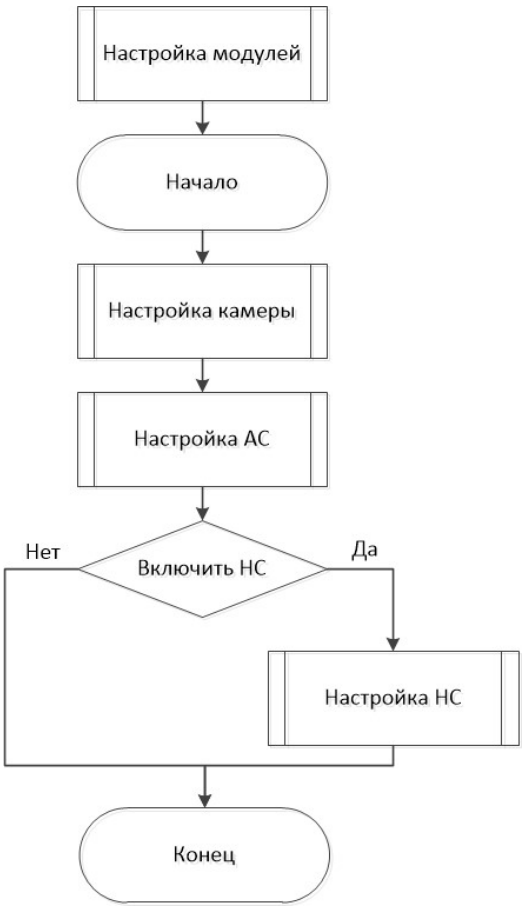

Рисунок 39 – Блок-схема функции настройки модулей

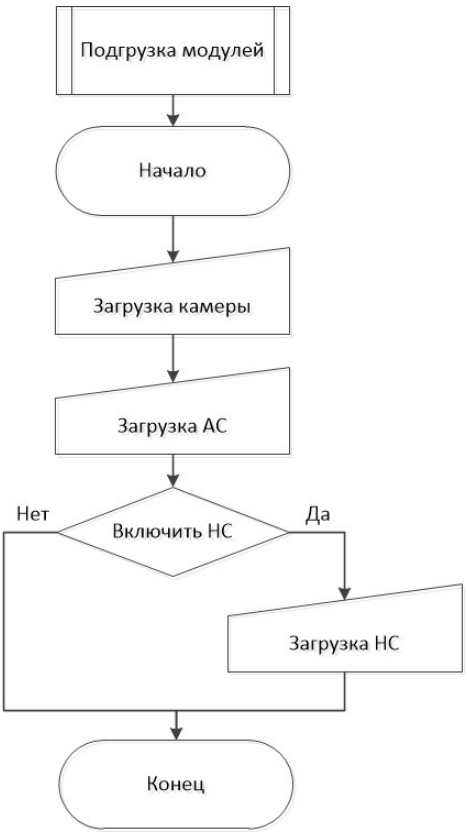

Рисунок 40 – Блок-схема функции подгрузки модулей

Приложение Ж. Таблица соответствия названий функций

Таблица 23. Таблица соответствия названия функций в блок-схемах и исходном коде программы

Модуль камеры	
calibrate	Калибровка камеры
draw_bbox_and_line	Отрисовка рамки и траектории
draw_bbox	Отрисовка рамки
draw_line	Отрисовка траектории
get_info	Вывод информации
get_img	Вывод изображения
setup	Настройка камеры
read_frame	Получение следующего камеры
calc_fps	Расчет FPS
exit_button	Проверка выхода
Модуль АС	
setup	Настройка АС
set_obj	Выбор объекта
update	Обновление положения объекта
Модуль НС	
setup	Настройка НС
detect	Распознавание объекта
Модуль Дисетчера	
__init__	Подгрузка модулей
setup	Настройка модулей
validate	Сравнение положения объекта с АС и НС
set_obj	Определение объекта
update	Обновление положения объекта

Приложение И. Диаграммы классов

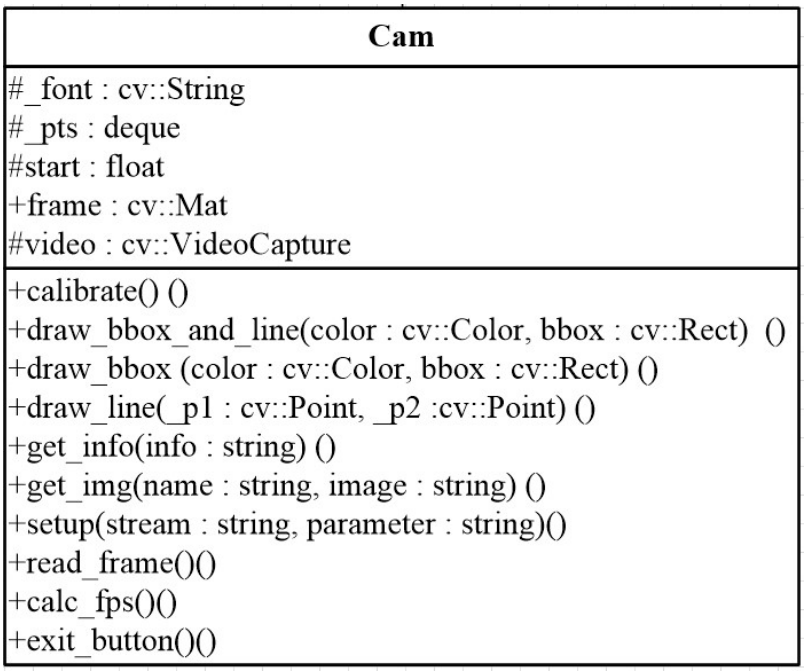

Рисунок 41 – Диаграмма класса Камеры

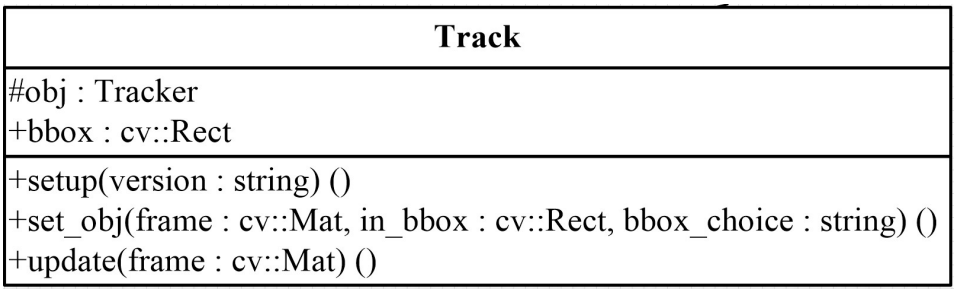

Рисунок 42 – Диаграмма класса АС

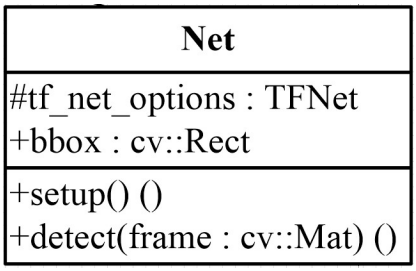

Рисунок 43 – Диаграмма класса НС

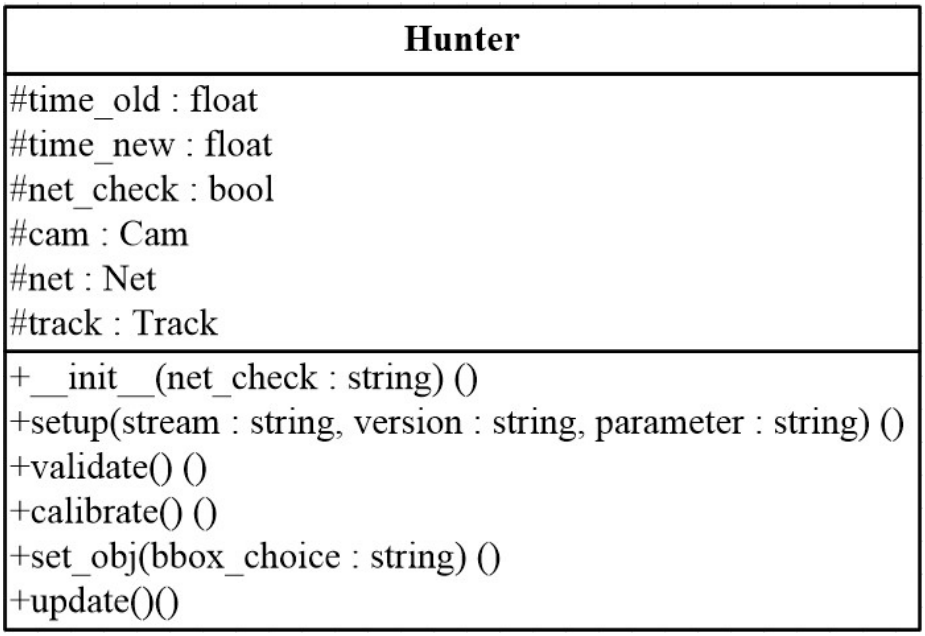

Рисунок 44 – Диаграмма класса Диспетчера

Приложение К. Исходный код программы

1) main.py

```
import sys
import hunter
import parser
# Parse input arguments
arguments_parser = parser.create()
arguments_list = arguments_parser.parse_args(sys.argv[1:])
# Init base flags and params
u_hunter = hunter.Hunter(arguments_list.net)
# Load all parameters, apply default constructors
u_hunter.setup(arguments_list.stream,
arguments_list.version,arguments_list.parameter)
# Calibrate camera if chosen
if arguments_list.calibration == 'on':
    u_hunter.calibrate()
# Set tracked object
u_hunter.set_obj(arguments_list.bbox)
# Update bbox and net check
flag = True
while flag:
    u_hunter.update()
```

2) cam.py

```
import time
from collections import deque
import cv2
import numpy as np
from numba import jit
class Cam:
```

```
# Simple font
_font = cv2.FONT_HERSHEY_SIMPLEX
# Store how object moves
_pts = deque(maxlen=64)
# Calculate FPS
start = time.time()
# Wait for camera calibration
# Interrupt by ESC key
def calibrate(self):
    flag = True
    while flag:
        self.read_frame()
        self.calc_fps()
        self.get_img('Calibrate', 'show')
        # Exit if ESC pressed
        flag = self.exit_button()
# Wrapper
def draw_bbox_and_line(self, color, bbox):
    # Draw bounding box
    _p1, _p2 = self.draw_bbox(color, bbox)
    # Draw line
    self.draw_line(_p1, _p2)
# Draw bbox in 3 possible colors
def draw_bbox(self, color, bbox):
    # Get corner values
    _p1 = (int(bbox[0]), int(bbox[1]))
    _p2 = (int(bbox[0] + bbox[2]), int(bbox[1] + bbox[3]))
    # Choose color
    if color == 'red':
        color = (0, 0, 255)
```

```
        elif color == 'green':
            color = (0, 255, 0)
        elif color == 'blue':
            color = (255, 0, 0)
        else:
            print('Wrong color')
        # Draw rectangle
        cv2.rectangle(self.frame, _p1, _p2, color, 2)
        return _p1, _p2
    # Draw track line
    def draw_line(self, _p1, _p2):
        # Loop over the set of tracked points
        for i in range(1, len(self._pts)):
            # If either of the tracked points are None,
            # ignore them
            if self._pts[i - 1] is None or self._pts[i] is None:
                continue
            # Otherwise, compute the thickness of the line and
            # draw the connecting lines
            thickness = int(np.sqrt(64 / float(i + 1)) * 2.5)
            cv2.line(self.frame, self._pts[i - 1], self._pts[i], (255, 0, 0), thickness)
        # Update the points queue
        self._pts.appendleft((int((_p1[0] + _p2[0]) / 2), int((_p1[1] + _p2[1]) /
2)))
    # Show debug info
    def get_info(self, info):
        # Present in separate window
        if info == 'gui':
            # Create black background
            _blank_image = np.zeros((250, 250, 3), np.uint8)
```

```
            # Add text
            # Processed FPS
            cv2.putText(_blank_image, 'FPS: '+ self.calc_fps(),(10, 30),
self._font)
            # Show info
            cv2.imshow('Info', _blank_image)
        # Output at console
        elif info == 'console':
            # Processed FPS
            print('FPS: ' + self.calc_fps())
        # Do not present
        elif info == 'off':
            pass
        else:
            print('False arguments.debug')
    # Output frame
    def get_img(self, name, image):
        # Choosing what to do with image
        # Save result
        if image == 'save':
            save = 'test/test.' + str(iter) + '.jpg'
            cv2.imwrite(save, self.frame)
        # Display result
        elif image == 'show':
            cv2.imshow(name, self.frame)
        # Display and save result
        elif image == 'save_and_show':
            # Save result
            save = 'test/test.' + str(iter) + '.jpg'
            cv2.imwrite(save, self.frame)
```

```
            # Display result
            cv2.imshow(name, self.frame)
        elif image == 'none':
            pass
        else:
            print('False arguments.image')
    # Open camera and setup tracker type
    def setup(self, stream, parameter=None):
        if stream == 'cam':
            self.video = cv2.VideoCapture(0)
        elif stream == 'video':
            self.video = cv2.VideoCapture(parameter)
        else:
            print('False arguments.stream')
        # If video not opened
        if not self.video.isOpened():
            print('cam.setup NOT OK')
    # Read frame and return it
    def read_frame(self):
        status, self.frame = self.video.read()
        if not status:
            status = 'cam.read_frame NOT OK'
            print(status)
    # Remember new point and return FPS
    @jit
    def calc_fps(self):
        end = time.time()
        seconds = end - self.start
        self.start = end
        return str(int(1 / seconds))
```

```
    # Return 0 if ESC button is pressed, 1 - otherwise
    def exit_button(self):
        k = cv2.waitKey(1) & 0xff
        if k == 27:
            return 0
        return 1
```

3) track.py

```
import cv2
class Track:
    # Setup tracker depends on version
    def setup(self, version):
        # Choose version
        if version == '310':
            # TLD tracker for OpenCV 3.1.0 and earlier
            self.obj = cv2.Tracker_create('TLD')
        elif version == '320':
            # TLD tracker for OpenCV 3.2.0 and later
            self.obj = cv2.TrackerTLD_create()
        else:
            print('False arguments.version')
    # Set tracked object
    def set_obj(self, frame, in_bbox, bbox_choice):
        # By coordinates or from net detection
        if (bbox_choice == 'coord') or (bbox_choice == 'net'):
            # Define an initial bounding box
            bbox = in_bbox
        # By person
        elif bbox_choice == 'roi':
            # Select Region Of Interest
            bbox = cv2.selectROI(frame, False)
```

```
            print("Object is selected")
        else:
            print("False arguments.bbox")
        # Initialize tracker with frame and bbox
        status = self.obj.init(frame, bbox)
        if not status:
            status = 'track.set_obj NOT OK'
            print(status)
    # Update object's bbox
    def update(self, frame):
        status, self.bbox = self.obj.update(frame)
        if not status:
            print('track.update NOT OK')
        return status
```

4) net.py

```
import time
import numpy as np
from darkflow.net.build import TFNet
class Net:
    # Set YOLO options
    def setup(self):
        options = {"model": "cfg/yolo.cfg", "load": "model/yolo.weights",
"threshold": 0.1}
        self.tf_net_options = TFNet(options)
    # Predict object class and return bbox in cup case
    def detect(self, frame):
        frame = np.asarray(frame)
        start_time = time.time()
        result = self.tf_net_options.return_predict(frame)
```

```
        end_time = time.time()
        result_time = end_time - start_time
        print('Detected at' + str(result_time) + ' s')
        for x in result:
            if x['label'].find('person') > -1:
                x0 = int(x['topleft']['x'])
                y0 = int(x['topleft']['y'])
                x1 = int(x['bottomright']['x'])
                y1 = int(x['bottomright']['y'])
                width = x1 - x0
                height = y1 - y0
                self.bbox = (x0, y0, width, height)
                print('Person is found')
```

5) hunter.py

```
import time
import cam
import net
import track
class Hunter:
    # Time to call net
    time_old = 0
    time_new = 0
    def __init__(self, net_check):
        if net_check == 'on':
            self.net_check = True
        elif net_check == 'off':
            self.net_check = False
        else:
            print('False net check')
```

```
        self.cam = cam.Cam()
        self.track = track.Track()
        if self.net_check:
            self.net = net.Net()
    def setup(self, stream, version, parameter=None):
        # Camera setup
        self.cam.setup(stream, parameter)
        # Setup tracker type and version
        self.track.setup(version)
        # Load net
        if self.net_check:
            self.net.setup()
    # Check bbox from tracker to bbox from classifier
    def validate(self):
        threshold = 20
        self.net.detect(self.cam.frame)
        if self.net.bbox is None:
            print('Object not found')
        if abs(self.track.bbox[0] - self.net.bbox[0]) > threshold or \
                    abs(self.track.bbox[1] - self.net.bbox[1]) > threshold or \
                    abs(self.track.bbox[2] - self.net.bbox[2]) > threshold or \
                    abs(self.track.bbox[3] - self.net.bbox[3]) > threshold:
            print("Bad tracker")
        else:
            print("Good tracker")
    def calibrate(self):
        self.cam.calibrate()
    def set_obj(self, bbox_choice):
        bbox = (50, 100, 200, 250)
        # Net detection
```

```
        if self.net_check:
            self.net.detect(self.cam.frame)
            bbox = self.net.bbox
        # Set object
        self.track.set_obj(self.cam.frame, bbox, bbox_choice)
    # Update bbox and call NN to verify object
    def update(self):
        # Read next video frame
        self.cam.read_frame()
        # Update tracker
        track_status = self.track.update(self.cam.frame)
        # Show info
        self.cam.get_info('gui')
        # Net checking
        if self.net_check:
            self.time_new = time.time()
            time_dif = self.time_new - self.time_old
            if time_dif > 10:
                print('Reset timer')
                self.net.detect(self.cam.frame)
                self.time_old = self.time_new
                # Reset object
                if not track_status:
                    self.track.set_obj(self.cam.frame, self.net.bbox, 'net')
        # Draw bounding box and track line
        self.cam.draw_bbox_and_line('red', self.track.bbox)
        # Present result
        self.cam.get_img('Tracking', 'show')
        # Draw bounding box
        self.cam.draw_bbox('green', self.net.bbox)
```

```
# Present result
self.cam.get_img('Net', 'show')
# Exit if ESC pressed
self.cam.exit_button()
```