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

Институт кибернетики
Направление подготовки 15.04.06 Мехатроника и робототехника
Кафедра интегрированных компьютерных систем управления

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

Тема работы
Разработка системы управления для робота-гексапода

УДК _____

Студент

Группа	ФИО	Подпись	Дата
8ЕМ41	Рудь Максим Николаевич		

Руководитель

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Профессор каф. ИКСУ	В.И. Гончаров	д.т.н.		

КОНСУЛЬТАНТЫ:

По разделу «Финансовый менеджмент, ресурсоэффективность и ресурсосбережение»

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент каф. МЕН	В.Ю. Конотопский	к.э.н.		

По разделу «Социальная ответственность»

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент каф. ЭБЖ	М.И. Пустовойтова	к.х.н.		

ДОПУСТИТЬ К ЗАЩИТЕ:

Зав. кафедрой	ФИО	Ученая степень, звание	Подпись	Дата
ИКСУ	А.В. Лиепиньш	к.т.н.		

Томск – 2016 г.

Запланированные результаты обучения по основной образовательной программе направления «Мехатроника и робототтехника»

В результате освоения программы магистратуры у выпускника должны быть сформированы общекультурные, общепрофессиональные и профессиональные компетенции.

Планируемы результаты обучения

Код рез-та	Результат обучения (выпускник должен быть готов)	Требования ФГОС, критериев и/или заинтересованных сторон
Профессиональные		
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.	Требования ФГОС (ПК-1, ПК-3, ОПК-1, ОПК-4, ОК-1, ОК-9), Критерий 5 АИОР (п. 1.1), согласованный с требованиями международных стандартов *EUR-ACE* и *FEANI*
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.	Требования ФГОС (ПК-3, ПК-4, ПК-7, ОПК-1, ОПК-3, ОК-1, ОК-4, ОК-5, ОК-6, ОК-9), Критерий 5 АИОР (пп. 1.1, 1.2), согласованный с требованиями международных стандартов *EUR-ACE* и *FEANI*
Р3	интегрировать и применять полученные знания для решения инженерных задач при разработке и проектировании современных мехатронных и робототехнических устройств и систем (в том числе интеллектуальных) с использованием достижений и технологий мирового уровня, современных инструментальных и программных средств.	Требования ФГОС (ПК-2, ПК-3, ПК-4, ПК-5, ПК-15, ПК-18, ОПК-3, ОПК-6, ОК-1, ОК-5, ОК-6, ОК-7), Критерий 5 АИОР (пп. 1.2), согласованный с требованиями международных стандартов *EUR-ACE* и *FEANI*
Р4	применять полученные знания для решения инженерных задач при производстве и эксплуатации современных мехатронных и робототехнических устройств и систем	Требования ФГОС (ПК-7, ПК-10, ПК-11, ПК-12, ПК-18, ОПК-4, ОПК-6, ОК-1, ОК-4, ОК-6, ОК-8), Критерий 5 АИОР (п. 1.3), согласованный с требованиями международных стандартов EUR-ACE и FEANI
Р5	планировать и проводить аналитические, имитационные и экспериментальные исследования для целей проектирования, производства и эксплуатации мехатронных и робототехнических средств и систем (в том числе интеллектуальных) с использованием передового отечественного	Требования ФГОС (ПК-1, ПК-2, ПК-3, ПК-4, ПК-5, ПК-6, ПК-13, ПК-17, ПК-18, ОПК-2, ОПК-3, ОК-1, ОК-3, ОК-4, ОК-6, ОК-7, ОК-8, ОК-9), Критерий 5 АИОР (п. 1.4), согласованный с требованиями международных

	и зарубежного опыта, уметь критически оценивать полученные теоретические и экспериментальные данные и делать необходимые выводы.	стандартов EUR-ACE и FEANI
Р6	понимать используемые современные методы, алгоритмы, модели и технические решения в мехатронике и робототехнике, знать области их применения, в том числе в гибких автоматизированных производствах.	Требования ФГОС (ПК-1, ПК-2 ПК-3, ПК-7, ОПК-1, ОПК-3, ОПК-4, ОК-5, ОК-9, ОК-10), Критерий 5 АИОР (п. 2.1), согласованный с требованиями международных стандартов EUR-ACE и FEANI
Универсальные		
Р7	эффективно работать в профессиональной сфере индивидуально и в качестве члена команды	Требования ФГОС (ПК-1, ПК-2 ПК-7, ПК-8, ПК-16, ПК-17, ОК-1, ОК-2, ОК-4, ОК-6, ОК-9), Критерий 5 АИОР (п. 2.1), согласованный с требованиями международных стандартов *EUR-ACE* и *FEANI*
Р8	владеть иностранным языком на уровне, позволяющем работать в интернациональной среде с пониманием культурных, языковых и социально-экономических различий	Требования ФГОС (ПК-4, ПК-8, ПК-9, ПК-16, ОПК-4, ОК-5), Критерий 5 АИОР (п. 2.2), согласованный с требованиями международных стандартов *EUR-ACE* и *FEANI*
Р9	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.	Требования ФГОС (ПК-5, ПК-8, ПК-15, ПК-16, ПК-18,ОПК-1, ОПК-4, ОПК-5, ОК-3, ОК-4, ОК-5, ОК-6, ОК-8, ОК-9), Критерий 5 АИОР (пп. 1.6, 2.3,), согласованный с требованиями международных стандартов *EUR-ACE* и *FEAN*
Р10	следовать кодексу профессиональной этики и ответственности и международным нормам инженерной деятельности	Требования ФГОС (ПК-8, ПК-11, ПК-16, ОПК-3, ОПК-6, ОК-4), Критерий 5 АИОР (пп. 2.4, 2.5), согласованный с требованиями международных стандартов *EUR-ACE* и *FEANI*
Р11	понимать необходимость обучения в течение всей жизни, уметь самостоятельно учиться и повышать свою квалификацию в течение всего периода профессиональной деятельности.	Требования ФГОС (ПК-4, ПК-8, ОПК-3, ОПК-4, ОК-5, ОК-6, ОК-7, ОК-8), Критерий 5 АИОР (2.6), согласованный с требованиями международных стандартов *EUR-ACE* и *FEANI.*

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

Институт кибернетики
Направление подготовки (специальность) Мехатроника и робототехника
Кафедра интегрированных компьютерных систем управления

УТВЕРЖДАЮ:
Зав. кафедрой
_____ _____ Лиепиньш А.В.
(Подпись) (Дата) (Ф.И.О.)

ЗАДАНИЕ
на выполнение выпускной квалификационной работы

В форме:

магистерской диссертации

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

Студенту:

Группа	**ФИО**
8ЕМ41	Рудь Максиму Николаевичу

Тема работы:

Разработка системы управления для робота-гексапода

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.).*	Объектом проектирования является система управления для класса шагающих мобильных роботов. Количество конечностей робота – от 4 до 8, количество степеней подвижности на одну конечность – от 2 до 6. Система управления должна решать две основные задачи: построение двухмерной карты окружающей местности, определяя местоположение робота на этой карте, и адаптацию походки робота к различным типам ландшафта. Задачи должны быть сведены в единый конвейер выполнения, обеспечивая управление в режиме «от датчиков к двигателям».

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. Аналитический обзор по литературным источникам с целью выяснения достижений мировой науки и техники в области управления шагающими роботами. 2. Выбор и синхронизация составляющих аппаратной части системы технического зрения (системы сенсоров, вычислительного блока, микроконтроллера, источника питания) 3. Разработка системы локализации и построения карты местности с использованием графического процессора 4. Интеграция разработанной системы в конвейер операционной системы Robot Operating System 5. Проведение теоретических исследований в области глубокого обучения с подкреплением и разработка алгоритма обучения походке для шагающего робота. 6. Обзор проведенных исследований с точки зрения экономической эффективности.
Перечень графического материала *(с точным указанием обязательных чертежей)*	

Консультанты по разделам выпускной квалификационной работы

(с указанием разделов)

Раздел	Консультант
Финансовый менеджмент, ресурсоэффективность и ресурсосбережение	доцент кафедры экономики, к.э.н., В.Ю. Конотопский
Социальная ответственность	доцент кафедры ЭБЖ, к.х.н., М.И. Пустовойтова

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

Разработка модуля ходьбы на основе обучения с подкреплением

Дата выдачи задания на выполнение выпускной квалификационной работы по линейному графику	10.10.2015

Задание выдал руководитель:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Профессор кафедры ИКСУ	В.И. Гончаров	д.т.н.		

Задание принял к исполнению студент:

Группа	ФИО	Подпись	Дата
8ЕМ41	Рудь Максим Николаевич		

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

Институт кибернетики
Направление подготовки 15.04.06 «Мехатроника и робототехника»
Уровень образования - магистр
Кафедра интегрированных компьютерных систем управления
Период выполнения - осенний / весенний семестр 2015/2016 учебного года

Форма представления работы:

Магистерская диссертация

(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)
22.05.2016	Основная часть	*60*
7.05.2016	Финансовый менеджмент, ресурсоэффективность и ресурсосбережение	*20*
8.05.2016	Социальная ответственность	*20*

Составил преподаватель:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Профессор	Гончаров В.И.	д.т.н.		

СОГЛАСОВАНО:

Зав. кафедрой	ФИО	Ученая степень, звание	Подпись	Дата
ИКСУ	Лиепиньш А. В.	к. т. н.		

Реферат

Магистерская диссертация состоит из 103 страниц, содержит 16 рисунков, 14 таблиц.

Ключевые слова: система управления, робот-гексапод, 3D-сканирование местности, локализация и построение карты, машинное обучение, глубокое обучение, обучение с подкреплением, нейронные сети, параллельные вычисления, CUDA.

Объект исследования: аппаратная и алгоритмическая части системы управления роботом – гексаподом.

Цель работы: разработка системы управления роботом – гексаподом.

Результат работы: спроектирована аппаратная и программная часть системы управления роботом гексаподом. Разработана и реализована система локализации и построения карты местности с использованием визуальной одометрии и вычислений на графическом процессоре. Разработана система глубокого обучения с подкреплением для шагающего робота.

Научная новизна полученных результатов заключается в применении глубокого обучения с подкреплением в задаче непрерывного управления шагающим роботом.

Область применения: разработанная система может применяться для управления широким классом шагающих роботов в решении задач инженерной разведки местности, переносе малогабаритных грузов по неровному ландшафту.

Определения, обозначения, сокращения, нормативные ссылки

GPU – графический процессор

GPGPU – использование графического процессора для вычислений общего назначения

CUDA – программно – аппаратная архитектура параллельных вычислений на графических процессорах, разработанная компанией NVIDIA

IT – информационные технологии

CPU – центральный процессор

API – интерфейс программирования приложений (набор готовых классов, процедур, функций, структур и констант, предоставляемых приложением для использования во внешних программных продуктах)

SDK — комплект средств разработки, который позволяет специалистам по программному обеспечению создавать приложения для определённого пакета программ

RGB – цветовая модель, в которой изображение состоит из трёх каналов – красного (R), зелёного (G) и синего (B)

RGB-D – тип изображения, в котором кроме цветовых каналов присутствует канал глубины (Depth).

SLAM – одновременная локализация и построение карты (simultaneous localization and mapping)

ROS – Robot Operating System, набор функций и библиотек для управления роботами

Оглавление

Введение

В настоящее время мобильные роботы используются для выполнения множества разнообразных задач. Их список включает в себя обслуживание складских помещений, перевозку грузов, исследование и мониторинг труднодоступных или опасных зон, разведку, взятие проб. Для этих целей было создан и исследован широкий спектр конструкций роботов. Последнее десятилетие во многих областях все большее применение находят шагающие роботы. Они обладают рядом очевидных преимуществ перед своими колесными или гусеничными аналогами. Шагающие роботы могут двигаться на неровном ландшафте и проникать в труднодоступные места благодаря возможности гибкого выбора модели передвижения. Также они могут сохранять дееспособность при потере одной или нескольких конечностей, перестраивая модель передвижения на ходу (при наличии соответствующей конструкции и алгоритмов управления). В данной работе исследуется робот с шестью конечностями – гексапод, который копирует строение и передвижение паука. Такая конструкция имеет все преимущества шагающих роботов, вместе с тем обладая более высокой стабильностью, нежели, например, антропоморфные роботы с двумя конечностями для передвижения.

Основной задачей при создании робота – гексапода является разработка алгоритма для его передвижения. Правильный алгоритм должен обеспечивать сохранение стабильности походки и положения туловища робота при максимально возможной на данном ландшафте скорости. Принимая во внимание разнообразие форм, типов, рельефов, консистенций возможных поверхностей передвижения, становится очевидным, что с использованием классических методов управления невозможно учесть все возможные состояния окружающей среды для того, чтобы под каждое из них вручную подобрать соответствующий алгоритм передвижения.

Для решения подобных задач с высокой степенью неопределенности во входных данных (в данном случае, в данных о состоянии окружающей среды) широко используются методы машинного обучения. Эти модели используют

большое количество данных для настройки своих параметров (обучение модели), для того чтобы затем применить полученные “знания” на новых данных (применение модели). Методы машинного обучения доказали свою высокую эффективность для решения задач классификации и регрессии, и в настоящее время используются во многих отраслях науки и техники. Подход, при котором для обучения модели используются промаркированные данные, называется *обучением с учителем*. Например, в задаче классификации изображений модель обучается на множестве примеров, каждому из которых соответствует метка принадлежности к определенному классу. Подход, при котором не используются маркированные данные, называется *обучением без учителя*.

Глубокое обучение – ветвь машинного обучения, которая находится на гребне волны методов решения такой проблемы, как распознавание изображений. «Глубина» моделей глубокого обучения идет от группирования функций в серии нелинейных трансформаций от входа, через промежуточные преобразования, к выходу. Общая композиция представляет собой глубокую модель, состоящую из слоев, и на каждом слое происходит очередной шаг от низкоуровневых деталей к высокоуровневым концепциям. Это приводит к иерархическому представлению проблемы восприятия. В данной работе предлагается использование каскада нескольких моделей глубокого обучения, каждая из которых обрабатывает некоторую часть данных о состоянии окружающей среды и внутреннем состоянии робота. Такой тип обучения называются мультимодальным. Две свёрточных нейронных сети используются для изучения высокоуровневых признаков ландшафта из RGBD-изображений, получаемых с инфракрасного сенсора глубины. Также модель учитывает данные о текущем состоянии конечностей робота. Эффективно решая задачу понижения размерности входных данных, изучаемые признаки помогут роботу выбирать правильную модель передвижения.

Обучение с подкреплением занимает промежуточную позицию между обучением с учителем и обучением без учителя. Его принцип заключается в том, что *агент* (в нашем случае, робот) обучается оптимальному поведению

непосредственно от взаимодействия с *окружающей средой*, делая вывод о степени полезности своих *действий*, получая *награды*. В данной работе предлагается использование модели обучения с подкреплением, называемой глубоким Q-обучением. В качестве входов используются данные о состоянии окружающей среды, а также данные о состоянии каждой из конечностей робота. Выходами модели являются значения полезности каждого из возможных действий (переходов между состояниями).

Помимо решения задачи ходьбы на неровном ландшафте, для осуществления инженерной разведки робот должен обладать возможностью построения карты окружающей местности и определять своё местоположение на ней. Эта задача получила название simultaneous localization and mapping (SLAM). При разработке системы SLAM для робота — гексапода в данной работе было скомбинировано два подхода. Тип карты, которую должен построить робот, был определен как occupancy grid (сетка занятости), которая показывает, какие области доступны для перемещения, а какие заняты препятствиями. В качестве математического аппарата был использован фильтр частиц (метод Монте — Карло). В качестве сенсора был использован Microsoft Kinect – RGBD-сенсор, работающий в режиме эмуляции дорогостоящего лазерного дальномера.

Чтобы получить необходимую точность локализации, необходимо охватить как можно больше вероятных состояний робота, что ведет к увеличению числа частиц в системе. Это делает применение фильтра частиц в реальном времени очень затратным с точки зрения скорости вычислений. Необходимо отметить, что большинство шагов фильтра выполняются независимо для каждой частицы, что позволяет осуществлять параллельную обработку всех частиц.

CUDA (Compute Unified Device Architecture) – программно-аппаратная архитектура параллельных вычислений на графических процессорах (GPU) компании NVIDIA. Это одна из наиболее распространенных технологий для осуществления вычислений общего назначения на GPU.

В данной работе на архитектуру CUDA был перенесен наиболее ресурсоемкий шаг фильтра частиц – вычисление весов, который выполняется независимо для каждой частицы и пригоден для параллельного исполнения. Для повышения наглядности полученных результатов алгоритм был реализован как на многоядерном CPU, так и на устройстве с поддержкой CUDA – мобильном чипе Tegra TK1.

1. Обзор литературы

Несколько последних десятилетий характеризуются быстрым развитием систем управления. Исследователи получили возможность устанавливать на шагающих роботов различные типы датчиков, а системы искусственного интеллекта начали активно применяться для анализа окружающей среды и движения роботов на сложных поверхностях. Множество работ посвящено применению генетических алгоритмов для синтеза походки для робота-гексапода [1], [2], [3]. Применение обучения с подкреплением также использовалось в нескольких работах [4], [5]. В [4] использовалась классическая методика Q-обучения для синтеза походки на ровном ландшафте. В [6] использовалась полносвязная нейронная сеть для обучения шестиногого робота походке. Также исследования проводились в области моделирования и классификации ландшафта для выбора оптимальной походки. В работе [7] использовался каскад из трех линейных классификаторов, которые выполняли свою работу, основываясь на информации различного рода – информации о текстуре, информации о рельефе (глубине) и информации о нагрузке на сервоприводах конечностей.

Использование глубоких нейронных сетей в задаче обучения с подкреплением началось в 2013 году с появлением работы [8] компании Google Deep Mind, использующей глубокую сеть для обучения компьютера играм компании Atari. С помощью этой технологии им удалось достичь сверхчеловеческих результатов. В 2015 году Deep Mind представила версию модели, победившую человека в игре Go!. Стоит заметить, что эти работы используют обучение с подкреплением в дискретном пространстве действий, то есть у агента есть только ограниченный набор доступных действий. Для робототехники больший интерес представляют модели, позволяющие осуществлять непрерывный контроль, то есть входящим данным о состоянии робота и окружающей среды ставить в соответствие конкретные значения углов поворотов моторов, каждое из которых – непрерывная величина. В этом направлении стоит упомянуть работу [9], в которой авторами предлагается

способ обобщения особой модели обучения с подкреплением - Q-обучения - на непрерывное пространство действий. В работе [10] авторами реализован алгоритм под названием «Управляемый поиск стратегии» (*guided policy search*). Суть этого метода заключается в «управлении» обучением нейронной сети с помощью классических алгоритмов оптимизации для поиска оптимальной траектории движения. Таким образом, нейронная сеть рассматривает только те варианты траекторий, которые находятся в некоторой окрестности оптимальной, найденной с помощью алгоритма оптимального контроля. Также в работе [16] рассматривается архитектура «Актер – Критик», которая позволяет применить Q-обучение в непрерывном состоянии действий путем использования нескольких глубоких нейронных сетей.

Активное использование свёрточных нейронных сетей для распознавания изображений началось около 5 лет назад. Появление графических процессоров для их обучения наряду с продвинутыми техниками, позволяющими повысить точность классификации на тестовой выборке, позволило свёрточным сетям достигнуть высочайших показателей в соревновании ImageNet [11]. В последние годы появилось множество архитектур, позволяющие использовать свёрточные сети не только для задач классификации, но также локализации и детектирования. Широкое распространение доступных по цене сенсоров глубины наподобие Microsoft Kinect, Intel RealSense, ASUS xTion, позволило создавать выборки RGBD-изображений и использовать их для обучения нейронных сетей. Существует несколько подходов к распознаванию RGBD-изображений. Некоторые из них используют операцию трехмерной свертки, как, например, в работе [12], но большинство применяют обычную операцию двухмерной свертки с использованием различных техник для обработки D-канала. Некоторые работы обрабатывают D-канал наряду с RGB. Эти подходы получили название 2.5D свёрточных сетей. Они демонстрируют хорошие результаты в задаче распознавания RGBD-изображений, однако интуитивно понятно, что они не принимают во внимание трехмерную структуру входных данных, а значит, теряют возможность изучения качественных признаков,

Добавлено примечание ([AO1]): реклама

Добавлено примечание ([AO2]): жесть-реклама

Добавлено примечание ([AO3]): это что такое? Пояснений нет

кодирующих геометрическую структуру сцены. Отдельного упоминания заслуживает применение свёрточных автокодировщиков для изучения высокоуровневых признаков в режиме без учителя. Автокодировщики – нейронные сети, вход которых равен выходу. Их задачей является понижение размерности входных данных. Автокодировщики могут рассматриваться как нелинейный аналог метода главных компонент. В работе [13] используется подобная модель для решения задачи предобучения свёрточной нейронной сети, но также подобные модели могут использоваться для фильтрации данных (фильтрующий автокодировщик) или для изучения признаков для дальнейшего их использования, например, в задаче обучения с подкреплением.

Методы решения задачи SLAM можно сгруппировать по типу используемых датчиков. Первая группа подходов основана на использовании связки лазерного дальномера и энкодеров двигателей мобильного робота. Наиболее успешные реализации этих методов представлены в работах [14], [15]. Вторая группа подходов использует две видеокамеры, объединенные в стереопару, что позволяет роботу ориентироваться в пространстве и строить его карту, детектируя и отслеживая с течением времени ключевые точки на входных изображениях. Третья группа методов возникла сравнительно недавно в связи с выходом на рынок недорогих RGBD-камер, которые не только выдают цветное изображение, но и соответствующую этому изображению карту глубины, в которой содержатся расстояния до всех точек.

В работе [16] технология CUDA использовалась для ускорения работы фильтра частиц. Авторами был сделан вывод, что наиболее дорогостоящим с точки зрения вычислений шагом являлся шаг отбора частиц. Ими было достигнутно ускорение в 5,73 раза по сравнению с реализацией на CPU.

В работе [17] была представлена система SLAM, основанная на ориентирах (landmarks). Авторы использовали GPU с поддержкой технологии CUDA для обработки видеоданных высокого разрешения.

Исходя из проведенного анализа литературы, можно сделать следующий вывод: применение глубокого обучения с подкреплением к не-игровым задачам

началось в 2014 году, и в области робототехники этот метод пока не исследовался для задач, отличных от манипуляции. Одними из наиболее эффективных систем SLAM являются системы, построенные на использовании фильтра частиц. Однако реализации подобных систем достаточно дороги за счет необходимости использования лазерного дальномера и ресурсоемки при использовании в фильтре большого количества частиц. Кроме того, в классических исполнениях они требуют использования одометров на колесах робота, что делает область их применения весьма ограниченной. В данной работе предлагаются пути устранения этих недостатков.

2. Анализ и сравнение колесного, гусеничного и шагающего способов передвижения

Хотя в природе не встречается колесного способа передвижения, ранние транспортные средства использовали колеса для движения благодаря изобретению паровых двигателей, железных дорог и двигателей внутреннего сгорания. Перемещение таким способом было очень удобно. Однако, когда в дело вступила неровная поверхность и неизвестные виды ландшафтов, колесные транспортные средства стали неподходящими. Необходимость решения этой задачи привела к изобретению гусеничного типа передвижения. У него также есть несколько недостатков, таких как разрушение поверхности движения по мере продвижения транспортного средства. Шагающий тип передвижения – это альтернатива первым двум типам, имитирующая шагающих животных в природе.

Наиболее важные элементы, определяющие движение и свойства робота – это характеристики ландшафта. Геометрия, материал и пространственные характеристики, такие как уклон, трение и жесткость, могут существенно различаться для разных видов ландшафтов. Необходимо определить характеристики ландшафта, задачу и критерии успешности выполнения и в

зависимости от этого выбрать характеристики робота. Преимущества и недостатки каждого из типов передвижения представлены ниже.

2.1. Колесное передвижение

Наиболее распространенный тип передвижения - это колесное, благодаря которому можно передвигаться плавно на ровных и рельефных ландшафтах. Колесные транспортные средства могут нести большой груз, энергоэффективность также можно считать преимуществом этого типа. Управление и разработка такого транспортного средства с технической точки зрения значительно проще, чем для других типов. Эта простота делает подобные системы наиболее распространенными на практике. Однако, когда в дело вступают неровные поверхности, препятствия или впадины, которые имеют больший радиус, чем колесо, устройство с этим типом передвижения не может действовать правильно. Например, в условиях песка или болот, использование колесного транспорта становится невозможным.

Добавлено примечание ([AO4]): -

2.2. Гусеничное передвижение

На неизвестных ландшафтах существенными преимуществами обладает транспорт с гусеничным типом передвижения. Гусеницы контактируют с землей на большой площади, что создает достаточное усилие для передвижения робота на разных типах поверхностей, таких как песок или снег. Гусеничный транспорт также относительно прост в разработке и может выдерживать большую полезную нагрузку. Однако, высокое энергопотребление можно отнести к основным недостаткам таких систем. Большая масса и потеря энергии на трение не позволяют назвать этот тип передвижения энергоэффективным. Другой недостаток – разрушение поверхности передвижения по пути следования. Этот тип транспорта использует трение скольжения, поэтому поверхность может быть существенно повреждена.

2.3. Шагающие системы

Шагающие системы – подходящее решение для неровных, рельефных или рыхлых поверхностей. Шагающие системы не контактируют с поверхностью на постоянной основе, в отличие от гусеничных и колесных платформ. Благодаря изолированным от друг друга конечностям, шагающие роботы имеют возможность выбирать правильные места для шага и движения. Изолированные конечности делают таких роботов очень эффективными, тогда как постоянный контакт с поверхностью для колесных и гусеничных решений лимитирует их эффективность, делая необходимым контакт с нежелательными частями поверхности передвижения. Шагающий робот может контролировать распределение нагрузки между ногами, вследствие этого имея активную подвеску. Независимо от характеристик ландшафта, шагающий робот может передвигать свое туловище в любом направлении.

Недостатками данного типа роботов является высокое энергопотребление, сложная кинематика, динамика и алгоритмы управления. Из-за того, что робот должен поддерживать баланс на постоянной основе, полезная нагрузка, которую робот может выдержать, сравнительно невелика. В шагающих роботах, моторы (наиболее массивные части) обычно приводят в движение каждый сустав, что делает конечности тяжелее, чем туловище, и приводит к низкому запасу по нагрузке. Стоит упомянуть об областях применения подобных роботов, которые включают в себя сбор материала с неизведанных поверхностей, спасательные операции после катастроф и в зонах, опасных для человека, вырубка деревьев в лесах и разминирование территории.

2.4. Статическая и динамическая стабильность

Стабильность – одна из основных характеристик анализа движения. Можно сказать, что стабильность – это поддержание роботом баланса. В общем случае, можно определить два вида стабильности: статическая и динамическая. Статическая стабильность определяется в случае, когда робот сбалансирован без применения дополнительной силы и без движения. Контакт ног с землей образует поддерживающий полигон и пока центр масс робота находится внутри этого полигона, робот статически стабилен.

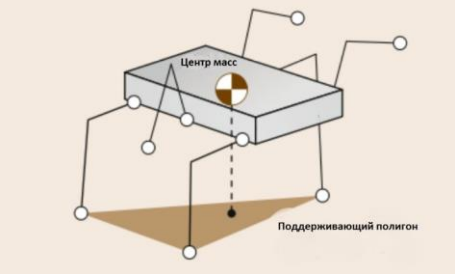

Рисунок 1 – Статическая стабильность гексапода

Динамическая стабильность требуется в то время, когда центр масс находится не внутри поддерживающего полигона. В этих случаях робот упадет, если не приложено дополнительных сил и нет движения.

Шестиногий шагающий робот имеет больший запас статической стабильности, чем четырехногий благодаря увеличению площади поддерживающего полигона.

2.5. Скорость передвижения робота

Другим фактором, определяющим передвижение робота, является скорость. Было показано, что скорость V_n (n – число конечностей) робота с

несколькими конечностями, в случае волнообразной походки, зависит от размера шага L_s , времени цикла T, и коэффициента использования β . β имеет прямое отношение к n. Фаза перехода t_T - время, когда нога в воздухе и перемещается на другую позицию $t_T = (1-\beta)T$ и позиционная фаза t_S - время, которое нога находится на земле, $t_S = \beta T$.

$$V_n = \frac{L_s}{T\beta} = \frac{L_s}{t_T}\left(\frac{1-\beta}{\beta}\right)$$

Робот с N конечностями имеет минимальный фактор использования $\beta_N = 3/\mathrm{N}$. Поэтому, скорость квадроподов, гексаподов и октоподов оперделяется как $V_4 = 0.33(L_S/T), V_6 = (L_S/T), V_4 = 1.6(L_S/T)$ соответственно. Можно сделать вывод, что чем больше конечностей имеет робот, тем большую скорость он может развить при волнообразной походке. Несколько типов походки для шагающих роботов рассмотрены в следущем разделе.

Существует множество типов конфигураций шестиногих роботов. В общем случае, гексаподы могут быть разеделены на два вида по конструкции: прямоугольные и шестиконечные. Прямоугольные варианты произошли от насекомых, конечности которых расположены по обеим сторонам туловища. Ноги шестиконечных гексаподов распределены равномерно по кругу туловища. Каждая нога может иметь от двух до шести степеней свободы, что также заимствовано от формы ног насекомых.

Прямоугольные варианты выглядят более натурально и заимствованы из природы. Эти роботы быстры в передвижении, но менее гибки в поворотах. Кроме того, несмотря на тот факт, что шестиконечные варианты необычны с точки зрения природы, они эффективны в передвижении во всех направлениях благодаря симметричности. Хотя варианты походок, применимые к прямоугольным конструкциям, могу быть также использованы для шестиконечных, они не всегда оптимальны или возможны.

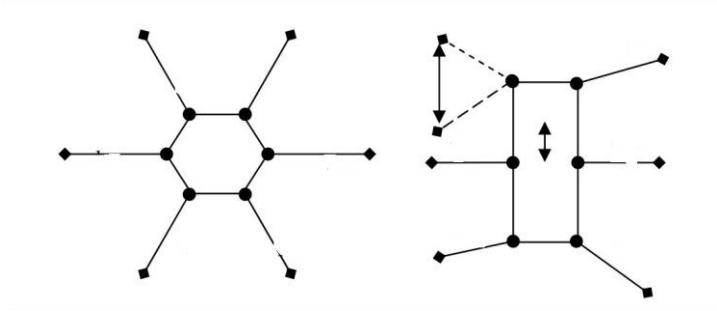

Рисунок 2 – Варианты исполнения гексаподов

Шестиконечные роботы имеют одинаковый размер шага во всех направлениях, тогда как прямоугольные разработаны для того, чтобы иметь большие шаги в направлениях вперед и назад. Статическая стабильность для обоих вариантов одинакова. Насекомые имеют разные последовательности движений ног, разные скорости и разные шаблоны передвижения. Во всех вариантах, статическая стабильность всегда соблюдается, что означает, что центр масс всегда находится внутри поддерживающего полигона. Анализ движения ноги может быть проведен в несколько стадий: фаза поддержки и фаза перехода (нога на земле и в воздухе соответственно). Принимая во внимание факт, что фаза перехода наиболее важна, разнообразные типы походок могут быть определены в зависимости от времени и последовательностей переходов. Некоторые походки, заимствованные из природы, приведены на рис. 3.

Насекомые используют разные походки, включая волнообразную – та, при которой движется только одна нога одновременно. Этот тип походки медленнее, чем та, при которой одновременно движется три ноги.

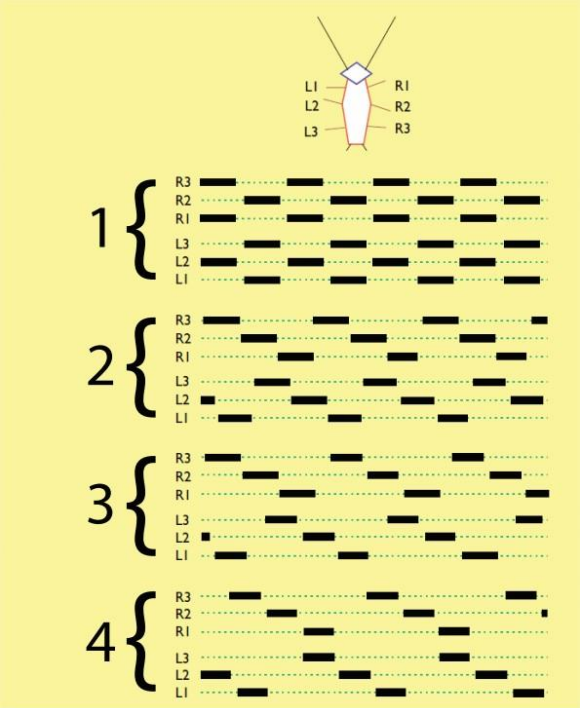

Рисунок 3 – Варианты походок для роботов-гексаподов

На рисунке 4 представлены 4 варианта походок, заимствованных от насекомых. Фазы перехода показаны черными прямоугольниками, пустое место между ними представляет фазу поддержки. Походка 1 – статически стабильный вариант «трипод», при котором одновременно движутся три ноги.

Добавлено примечание ([AO5]): А остальные походки? Я так понимаю, их эффективность меньше – большее время на опоре, меньшее число ног в воздухе

3. Проектирование и разработка системы управления

3.1. Определение задач, решаемых системой управления

В данном пункте определим, какие основные задачи должна решать система управления роботом – гексаподом для использования его в таких сферах, как инженерная разведка местности и доставка малогабаритных грузов по неровному ландшафту.

Добавлено примечание ([AO6]): Это уже не число задачи. Это задачи сразу с решением. Как будто обзора вариантов не проводилось, а сразу были выбраны решения.

Добавлено примечание ([AO7]): Это в заголовке написано =)

1) Обеспечение стабильной походки на ландшафтах с разнообразной

структурой, текстурой и рельефом. В данной работе для решения этой задачи предлагается использование глубокого машинного обучения с подкреплением.

2) Построение карты местности и определение своего местоположения на ней. В данной работе используется система SLAM, основанная на фильтре частиц. В первом прототипе разработки в качестве основного сенсора для получения видеоинформации и информации о дальности объектов используется RGB-D сенсор Microsoft Kinect. Он же используется для получения одометрии (метод визуальной одометрии).
3) Интеграция двух первых модулей с модулем планирования пути к цели. Для решения задачи навигации в данной работе используется алгоритм A*.
4) Обеспечение взаимодействия первых трех модулей для реализации типа управления «от сенсоров к двигателям». Для этого в работе используется операционная система Robot Operating System, которая предоставляет систему взаимодействия между основными узлами типа «подписчик – публикующий».

3.2. Аппаратное обеспечение системы управления

3.2.1. Выбор сенсоров

1) Для решения задачи визуальной одометрии, а также для получения данных о текстуре ландшафта роботу необходима видеокамера.
2) Для получения информации о глубине сцены, местонахождении объектов, определения расстояния до следующей контрольной точки движения, а также построения карты местности, роботу необходимы датчики расстояния, которые могут быть ультразвуковыми, лазерными и инфракрасными.

В данной работе для создания прототипа системы управления было решено использовать комбинированный сенсор Microsoft Kinect.

Сенсор Kinect включает в себя RGB – камеру, датчик глубины, который состоит из инфракрасного излучателя и инфракрасного КМОП – сенсора.

Инфракрасный излучатель и проектор работают вместе для получения информации о глубине сцены перед сенсором. Таким образом, с данного сенсора может быть получено 2 изображения – цветное изображение разрешением 640 x 480 точек, и карта глубины в градациях серого того же разрешения (2048 различных значений, от 2048 – белый до 0 – черный).

Техника, используемая в сенсоре Kinect, называется структурированным световым трехмерным сканированием. Эта техника используется во многих промышленных проектах, таких как контроль качества и измерение объема подразумевает использование высокоточных и дорогих сенсоров. Kinect – первый сенсор для широкой аудитории, который использует эту технику. [17]

В таблице 1 приведено сравнение сенсоров подобного типа от разных производителей по нескольким важным критериям:

Таблица 1 - Сравнение характеристик видеосенсоров

Сенсор	3D разрешение, px	RGB, px	Частота, fps	ПО	Цена
PrimeSense Carmine	640x480, 320x240, 160x120	640x480, 320x240.	15, 30.	OpenNI	6000
Microsoft Kinect	640x480, 320x240	640x480, 320x240.	15, 30.	Microsoft Kinect SDK OpenNI	4000
Microsoft Kinect v2	512 x 424	1920 x 1080	15, 30	OpenNI	10000
Стереокамера	До 1920 x 1080	До 1920 x 1080	До 30	OpenCV	~3000

Таким образом, после рассмотрения основных характеристик данного сенсора, можно сделать следующие выводы:

1) Комбинация сенсора глубины и видеокамеры идеально подходит для решения поставленных задач на стадии создания прототипа в лабораторных условиях.

2) Стоимость сенсора (4000 рублей) значительно ниже, чем у высокоточных лазерных дальномеров, которые используются в профессиональных проектах.
3) Чип PS1080 избавляет от необходимости самостоятельно интерпретировать получаемые данные. На входе мы имеем сетку с готовыми значениями глубин каждой точки по шкале от 0 (максимально близко) до 2048 (максимально далеко).

Стоит отметить, что выбранный сенсор подходит именно для разработки прототипа для тестирования в лабораторных условиях, так как позволяет получать и интерпретировать данные без разработки дополнительного программного обеспечения и обладает низкой стоимостью. Однако, для применения робота в реальных условиях он не подходит, так как инфракрасный дальномер не работает в условиях открытой местности. Кроме того, качество изображения с видеокамеры недостаточно высоко для того диапазона цветов, который наблюдается на открытых пространствах, например, в солнечную погоду. В связи с этим, на дальнейших стадиях разработки будет добавлена возможность получения карты глубины от стереокамеры.

2.2.2 Выбор бортового компьютера

В связи с тем, что робот должен функционировать полностью автономно, все необходимые вычисления должны проводиться на борту. Вычислительный блок должен удовлетворять следующим условиям:

1) Должен иметь порт USB для подключения сенсора Kinect.
2) Должен иметь возможность подключения к интернету для передачи информации по беспроводному каналу связи.
3) Должен иметь линейные размеры не более 30 см x 30 см x 5 см и массу не более 1 кг ввиду особенностей конструкции робота-гексапода (стоит задача минимизации массы аппаратной части системы управления).
4) Должен иметь графический процессор с поддержкой технологий CUDA
5) Должен иметь операционную систему семейства Windows или Unix.

6) Должен обладать максимально возможной энергоэффективностью для обеспечения длительного времени автономной работы робота.
7) Должен иметь стоимость не более 500 $.

Рассмотрим основные варианты бортовых вычислительных станций, представленных на рынке.

1) Ноутбук или нетбук. Большинство подобных устройств, имеющих малые размеры и массу, обладают низкопроизводительным центральным процессором типа Intel Celeron или Intel Pentium и встроенным графическим процессором, что не подходит для исполнения ресурсоемких вычислений при обработке изображений и применении глубоких моделей машинного обучения в режиме реального времени. Кроме того, ноутбуки и нетбуки имеют дисплей, который создает лишнюю массу и потребляет дополнительные ресурсы.
2) Raspberry Pi. Плюсами данной платформы является ее низкая стоимость (35$) и наличие операционной системы, удовлетворяющей поставленным условиям. Минусы – низкая производительность, малый объем оперативной памяти, отсутствие CUDA-совместимого графического процессора.
3) Cubieboard A20. Построен на базе мощного двухъядерного процессора ARM Cortex A7, обладает 1 Гб оперативной памяти DDR3, всеми необходимыми интерфейсами и операционной системой Ubuntu. Стоимость – 150 $.
4) NVIDIA Jetson TK1. В марте 2014 года на конференции GPU Technology Conference компанией NVIDIA был представлен мобильный суперкомпьютер Jetson TK1. Основан на базе системы-на-чипе NVIDIA Tegra K1 – первой мобильной платформы, которая обеспечивает полную поддержку технологий CUDA, OpenGL, DirectX 11. Tegra K1 является комбинацией 192 вычислительных ядер CUDA с новейшей архитектурой Kepler и четырехъядерного процессора ARM Cortex A15.

Jetson TK1 обладает 2 Гб оперативной памяти и модулем памяти на 16 Гб, соответствующим спецификации eMMC. Имеет все необходимые интерфейсы и поддерживает операционную систему Linux. Стоимость устройства 192 $ [18].

5) X210ii Package C – одноплатный компьютер на базе микроконтроллера SAMSUNG S5PV210. Отличительными особенностями являются наличие высокопроизводительного процессора Cortex-A8 с тактовой частотой 1ГГц, 512 Мб оперативной памяти и 4 Гб flash – памяти. Поддерживает операционные системы Linux и Android. Имеет в составе сенсорный LCD – экран. Присутствует графический чип PowerVGR SG, который поддерживает последние версии графических API OpenGL, DirectX. Подобные чипы включены в несколько популярных портативных устройствах, таких как Apple iPhone и Apple iPad. Стоимость компьютера – 14500 рублей.

В таблице 1 представлены результаты сравнения бортовых вычислительных станций по основным параметрам.

Таблица 2 - Сравнение компьютеров

	Lenovo IdeaPad S210	**Raspberry Pi**	**Cubieboard A20**	**Jetson TK1**	**X210ii Package C**
Процессор	Intel Celeron 1,5 ГГц	Broadcom BCM2835 700 МГц	ARM Cortex A7 1 ГГц	ARM Cortex A15 2,3 ГГц	ARM Cortex A8 1 ГГц
Оперативная память	2 Гб	256 Мб	512 Мб	2 Гб	512 Мб
Хранилище данных	320 Гб	внешнее	4 Гб	16 Гб	4 Гб
Доступ в интернет	да	да	да	да	да
Необходимая ОС	да	да	да	да	да
Наличие графического процессора	да	нет	нет	да	да
Поддержка CUDA	нет	нет	нет	да	нет

Стоимость (руб)	20000	2900	48000	13000	14500

Исходя из требований к бортовой вычислительной станции, а также принимая во внимание ресурсоемкость предстоящих вычислений и необходимость высокой энергоэффективности, из представленных вариантов выбираем мобильный суперкомпьютер NVIDIA Jetson TK1 по следующим параметрам

1) Имеет приемлемые массогабаритные характеристики.
2) Имеет высокое соотношение цена / (производительность и характеристики).
3) Единственная мобильная платформа, имеющая поддержку CUDA.
4) Обладает емким накопителем
5) Имеет подходящую операционную систему.

2.2.3 Выбор микроконтроллера

Для передачи сигналов двигателям исполнительных механизмов робота, а также сбора информации с датчиков других типов, таких как датчики температуры, освещенности, задымленности, анализаторы газа, акселерометры и гироскопы, потребуется микроконтроллер, который будет обмениваться информацией с компьютером Jetson TK1. Выберем микроконтроллер, который будет соответствовать следующим требованиям:

1) Присутствует возможность интеграции с основным программным обеспечением
2) Программируется на языке высокого уровня
3) Обеспечивает необходимый функционал

В качестве микроконтроллера был выбран Arduino Mega по нескольким параметрам:

4) Большое количество доступных плат расширения и датчиков.
5) Простота реализации обмена информацией с основной программой.
6) С-подобный язык программирования.

Итогом данной части работы стал обоснованный выбор аппаратного обеспечения прототипа системы управления роботом – гексаподом. В качестве основного источника визуальной информации используется RGBD-сенсор Kinect, обеспечивающий приемлемое качество входных данных, простоту их интерпретации, а также поддерживающий работу в среде Linux. Для реализации вычислений на борту робота используется одноплатный компьютер Jetson TK1, обладающий поддержкой технологии CUDA, используемой для работы основных модулей системы управления роботом. Для непосредственного управления сервоприводами конечностей используется микроконтроллер Arduino Mega, который обменивается данными с основной программой через последовательный порт.

3.3. Разработка модуля локализации и построения карты

3.3.1. Алгоритм

Методы решения данной задачи можно сгруппировать по типу используемых датчиков. Первая группа подходов основана на использовании связки лазерного дальномера и энкодеров двигателей мобильного робота. Вторая группа подходов использует две видеокамеры, объединенные в стереопару, что позволяет роботу ориентироваться в пространстве и строить его карту, детектируя и отслеживая с течением времени ключевые точки на входных изображениях. Третья группа методов возникла сравнительно недавно в связи с выходом на рынок недорогих RGBD-камер, которые не только выдают цветное изображение, но и соответствующую этому изображению карту глубины, в которой содержатся расстояния до всех точек.

Недостатками первой группы методов можно считать применимость только для колесных платформ, сильную зашумленность колесных энкодеров и дороговизну лазерных дальномеров. Подходы, использующие изображения с камер, сильно зависимы от набора ключевых точек на изображениях и плохо работают в визуально бедных средах. Инфракрасные RGBD — камеры не работают на открытых пространствах.

При разработке системы SLAM для робота — гексапода было скомбинировано два подхода. Тип карты, которую должен построить робот, был определен как occupancy grid (сетка занятости), которая показывают, какие области доступны для перемещения, а какие заняты препятствиями. В качестве математического аппарата был использован фильтр частиц (метод Монте — Карло). В качестве сенсора был использован Kinect, работающий в режиме эмуляции дорогостоящего лазерного дальномера.

Доступность относительно недорогих лазерных дальномеров дало толчок к исследованиям применимости сенсоров этого типа к решению задачи SLAM. С течением времени сформировались две группы методов, которые различаются по используемому математическому аппарату. Первая группа методов использует расширенный фильтр Калмана (EKF), вторая же группа – фильтр частиц (Particle filter). В данной работе мы подробно остановимся на второй группе методов.

Фильтр частиц представляет собой моделирующий метод оценки состояния не полностью наблюдаемой системы. Он хранит взвешенное, нормализованное множество выборки состояний $S = \{s_1, s_2 ... s_n\}$, называемых частицами. На каждом шаге, получив вектор измерений z, фильтр:

1. Совершает переход в новое состояние в Марковской модели позиции робота $p(s_i / s_{i-1})$. Это действие моделирует движение робота в пространстве.
2. Взвешивает каждое полученное состояние согласно Марковской модели наблюдений $p(z / s_i)$.
3. Создает выборку m новых состояний S' из S с заменой.
4. Нормализует веса для нового множества состояний.

Фильтр частиц хорошо подходит для решения задачи локализации, где необходимо отслеживать позицию робота, которая является скрытой величиной. Переход между состояниями – это движение робота, а наблюдения – показания лазерного дальномера. Обе эти величины сильно зашумлены.

Изменение внутреннего состояния с течением времени производится посредством моделирования движения внутри робота. Обычно в качестве базиса модели движения выбирается перемещение, измеренное системой одометрии. Одометрия может очень точно измерять угол поворота колес, однако определить реальные перемещения робота с ее помощью проблематично. Проскальзывания, сдвиги и неровности подстилающей поверхности могут стать причиной ошибок, которые со временем будут только накапливаться. Модели движения для различных роботов на различных типах поверхностей могут серьезно отличаться, но все они будут тем или иным образом учитывать систематические и случайные ошибки. Рассмотрим простейшую модель движения. Пусть, согласно показаниям одометра текущее положение робота изменилось на x, y, θ . Фильтр частиц применяет модель ошибки и для каждой созданной частицы i получает:

$$x_i = a_x \cdot x + b_x + N(0, \sigma_x)$$
$$y_i = a_y \cdot y + b_y + N(0, \sigma_y)$$
$$\theta_i = a_\theta \cdot \theta + b_\theta + N(0, \sigma_\theta)$$

Переменные a и b представляют собой систематические ошибки определения положения при движении. Функция $N(0, \sigma)$ добавляет случайный шум с нормальным распределением вблизи точки 0 и имеющим стандартное отклонение σ. Стандартное отклонение определяется для каждого робота индивидуально экспериментальным путем и может зависеть от множества факторов.

После моделирования необходимо взвесить все частицы согласно текущим наблюдениям робота. При решении проблемы чистой локализации, робот имеет полную карту в памяти. Позиция, описанная каждой частицей, соответствует некоторой известной точке и направлению на карте. Следовательно, можно сравнительно легко определить, какие показания должен вернуть дальномер, если робот находится в этой позиции. Обычно предполагают, что погрешность показаний дальномера имеет нормальное распределение, и, следовательно, если, согласно карте, первое препятствие должно встретиться на расстоянии d, а

показание датчика d' , то ошибка $\delta = d - d'$ будет распределена нормально вокруг точки 0. Предполагая, что значение позиции робота и показания дальномера можно рассматривать как независимые случайные величины, полный вес частицы *i* будет иметь вид:

$$P_i = \prod_k P(\delta_{ik} | s_i, m)$$

,

где δ_{ik} - разница между ожидаемым и измеренным расстоянием до препятствия на проходе лазера k в частице (положении) *i*.

Согласно Murphy [4], ключевая идея так называемого фильтра частиц Рэо – Блэквелла (Rao-Blackwellized particle filter) для решения задачи SLAM – оценка апостериорной вероятности $p(x_{1:t}, m / z_{1:t}, u_{1:t-1})$ структуры карты и траектории $x_{1:t} = x_1, ..., x_t$ робота. Эта оценка происходит с использованием измерений $z_{1:t} = z_1, ..., z_t$ и показаний одометра $u_{1:t-1} = u_1, ..., u_{t-1}$. Фильтр частиц Рэо-Блэквелла использует следующее математическое выражение:

$$p(x_{1:t}, m \mid z_{1:t}, u_{1:t-1}) = \\ p(m \mid x_{1:t}, z_{1:t}) \cdot p(x_{1:t} \mid z_{1:t}, u_{1:t-1}).$$

Это позволяет нам сначала оценить только траекторию робота и затем вычислить карту, используя эту траекторию. Поскольку карта сильно зависит от оценки положения робота, этот подход решает проблему. Если известна позиция робота, то составление карты, используя технику картирования с известными позициями, не составляет труда.

Чтобы оценить апостериорную вероятность $p(x_{1:t} \mid z_{1:t}, u_{1:t-1})$, можно применить фильтр частиц. Каждая частица представляет собой потенциальную траекторию робота. Карта строится исходя из измерений сенсора и траектории, которую представляет собой каждая частица. Одним из самых известных алгоритмов фильтрации с помощью частиц является фильтр по технике набор – взвешивание – новый набор. Фильтр частиц Рэо – Блэквелла для построения карты обрабатывает данные одометрии и измерения сенсоров, как только они

поступают на вход. Фильтр обновляет набор частиц, которые представляют апостериорное распределение. Этот процесс может быть описан следующими шагами:

1. Карту окружающего пространства представляем двумерным массивом однобайтовых переменных. В начальный момент карта пуста.
2. Инициализируем начальное положения робота.
3. Считываем значения с дальномера и наносим на карту препятствия согласно полученным измерениям.
4. Далее идёт работа в бесконечном цикле.
    1. Считываем с одометрии, на сколько переместился робот относительно предыдущего измерения.
    2. Осуществляем переход внутри фильтра согласно полученным данным.
    3. Считываем показания с дальномера.
    4. При помощи фильтра частиц, используя текущую карту, вычисляем наиболее вероятное положение робота. Одна частица в фильтре частиц содержит положение и угловую ориентацию робота. Вероятность частиц рассчитывается на основе разности реальных показаний дальномера и показаний, которые должны были бы быть в данной частице.
    5. Обновляем карту.
    6. Создаем новый набор частиц, куда попадают частицы с наивысшими вероятностями.

3.3.2. Получение вектора измерений с сенсора Kinect

Как было отмечено ранее, вес частицы зависит от того, насколько точно показания «виртуальных» датчиков частицы совпадают с реальными показаниями сенсора Kinect. Исходя из этого, необходимо создать некую

аппроксимирующую модель сенсора Kinect, которой могли бы пользоваться частицы для получения своих показаний.

Карта глубины, получаемая сенсором, размера 640 x 480 точек, содержит значения от 0 до 255. 255 – наиболее удаленный объект. Для создания модели сенсора карта глубины разбивается на 36 регионов.

Для каждой ячейки размера 107 x 80 находится среднее значение глубины путем сложения значений всех пикселей и деления полученной суммы на общее число пикселей. Полученная матрица размера 6 x 6 разбивается на столбцы. Для каждого из 6 столбцов находится минимальное значение.

Таким образом, мы получаем своеобразный набор из 6 дальномеров, лучи которых расположены по дуге с промежутком 10 градусов. Подобная модель будет использована частицами для получения расстояний до объектов на карте.

3.3.3. Получение показаний визуальной одометрии

Для получения показаний одометрии из визуальной информации используется механизм ключевых точек, совмещенный с получением показаний инфракрасного дальномера. Основные шаги алгоритма представлены ниже:

1. Детектирование ключевых точек и расчёт дескрипторов для них с помощью алгоритма SURF. Операция проводится для двух последующих изображений.
2. Выбор нескольких ключевых точек для анализа. При выборе применяется фильтр расстояния (точки не должны располагаться слишком близко друг к другу)
3. Производится сопоставление ключевых точек на последующих кадрах путем вычисления евклидова расстояния между дескрипторами.
4. Расстояние, пройденное роботом, вычисляется как разница между текущим показанием дальномера в соответствующей точке и предыдущим. Далее находится среднее арифметическое между показаниями для всех ключевых точек.
5. Угол поворота рассчитывается стандартными средствами библиотеки OpenCV с использованием оптического потока.

3.3.4. Механизм отбора частиц

1) Нормализация весов. Веса всех частиц складываются, вес каждой частицы делится на полученную сумму.
2) Построение нового набора частиц осуществляется путем использования *колеса отбора* (рис.3):

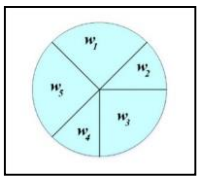

Рисунок 4 - Колесо отбора

Частицы с большим весом занимают соответственно большую долю колеса, и наоборот.

Создадим переменную *index* и придадим ей случайное значение в интервале от 0 до N, где N – количество частиц. Создадим также переменную β , инициализировав её нулем. Теперь, в цикле от 0 до N выполним следующие действия:

для всех *i* от 0 до N

$\beta = \beta + rand(0...2 \cdot w_{max})$ // w_{max} - максимальный вес частицы в наборе

если (частица[index].*w* < β)

{

$\beta = \beta -$ частица[index].*w*

index = index + 1;

}

иначе

{

Поместить частицу в новый набор

}

конец

3.3.5. Модель вычислений CUDA

Технология CUDA реализует модель вычислений типа «сетка». Низшими звеньями в этой модели выступают так называемые «нити», которые выполняют элементарные операции, например, сложение элементов двух векторов. Нити объединяются в блоки, причем мы можем пользоваться тремя измерениями. На практике обычно используются одномерный или двухмерный случаи. Блоки, в свою очередь, объединяются в сетку (*grid*), которая имеет два измерения.

3.3.6. Предсказание следующего положения робота

Для реализации данного шага на GPU мы должны запустить ядро (специальное название для функции, выполняемой на GPU), с количеством нитей равным количеству частиц. Аргументами ядра выступают предыдущая позиция робота и показания одометрии. Случайные числа для реализации шума генерируются непосредственно на GPU с помощью библиотеки *cuRAND*, поставляемой вместе с *CUDA Toolkit*.

Каждая нить ядра выполняет все необходимые вычисления для соответствующей ей частицы.

3.3.7. Моделирование сенсора Kinect

Карта глубины с Kinect поступает на вход ядра CUDA, запущенного для тридцати шести двухмерных блоков размера 107 x 80 нитей. Каждому блоку соответствует свой регион карты глубины. Для каждого блока обрабатываемый участок изображения копируется в разделяемую память для более быстрого доступа к данным. Далее, с помощью параллельной редукции вычисляется сумма всех элементов в блоке и делится на количество пикселей. Полученное значение записывается в 36-мерный вектор в глобальной памяти GPU. После того, как все блоки завершили работу, полученный вектор с найденными

средними значениями секторов подается на вход второго ядра, запущенного для 1 блока из 36 нитей. Данное ядро, также с помощью параллельной редукции, вычисляет максимумы среди шести элементов каждого из столбцов. Полученный в результате вектор из шести элементов поступает на вход ядра, вычисляющего веса частиц.

3.3.8. Вычисление весов

Данный шаг реализуется на GPU аналогично предыдущему. Мы вновь запускаем ядро, где каждая нить отвечает за вычисление веса для соответствующей частицы в наборе. На вход ядра поступает вектор данных с сенсора Kinect, а также карта помещения, размещенная в текстурной памяти GPU. После проведения вычислений, массив с весами частиц передается в оперативную память для проведения отбора, который на данной стадии работы выполняется полностью на CPU. Это объясняется тем, что наиболее сложная часть алгоритма (непосредственно отбор) не обладает параллелизмом. Проведенные тесты показали, что прирост производительности, полученный при выполнении отдельных шагов отбора, таких как нормализация весов и вычисление средних значений переменных, практически полностью нивелируется временем копирования данных между CPU и GPU.

Для наглядности алгоритм был протестирован на нескольких значениях количества частиц в системе. График производительности алгоритма при реализации на CPU и на GPU представлен на рис.5:

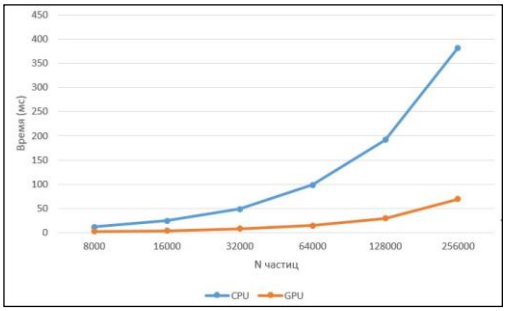

Рисунок 5 - График производительности алгоритма Монте-Карло для решения задачи одновременной локализации и построения карты

N частиц	Предсказание (мс)	Вычисление весов (мс)	Отбор (мс)	Нахождение среднего (мс)	В общем (мс)
8000	2,62	8,58	0,08	0,64	11,92
16000	5,33	17,41	0,18	1,3	24,22
32000	10,6	34,54	0,38	2,75	48,27
64000	20,75	71,34	0,89	5,12	98,1
128000	41,76	137,46	1,83	10,39	191,44
256000	82,56	271,75	6,98	20,49	381,78

Рисунок 6 – Время исполнения шагов алгоритма на CPU

N частиц	Предсказание (мс)	Вычисление весов (мс)	Отбор (мс)	Нахождение среднего (мс)	В общем (мс)
8000	0,57	0,69	0,08	0,64	1,98
16000	1,12	1,25	0,18	1,3	3,85
32000	2,22	2,34	0,38	2,75	7,69
64000	4,42	4,31	0,89	5,12	14,74
128000	8,91	8,22	1,83	10,39	29,35
256000	17,7	16,14	6,98	20,49	69,31

Рисунок 7 – Время исполнения шагов алгоритма на GPU

3.4. Разработка модуля ходьбы на основе обучения с подкреплением

Методы обучения с подкреплением (Reinforcement learning) находят свое применение как в теории управления, так и в когнитивных науках. В теории управления, построение идеально работающей системы проблематично, а порой и невозможно, когда на систему действуют непредвиденные возмущения. Но при создании системы, которая учится выполнению задачи самостоятельно, разработка сложных алгоритмов управления перестает быть необходимой. В

когнитивных науках, способность учиться – ключевой компонент восприятия. В данном разделе рассматривается способность робота – гексапода научиться ходить, используя лишь информацию о состоянии собственных приводов, а также о состоянии окружающей среды в виде визуальных данных. В данном случае робот может рассматриваться как биологический субъект, помещенный в среду и лишенный всякого супервизорного контроля.

В случае обучения с учителем, цели, т.е. правильные ответы, даются агенту на протяжении обучения. В некоторых задачах управления сложно, а порой и невозможно определить конкретные «правильные» решения, которые агент должен повторять. В обучении с подкреплением, вместо конкретных правильных ответов агент получает от среды награду, которая показывает, правильные он предпринимает действия или нет. Задачей обучающего алгоритма будет нахождение таких действий, выполнение которых ведет к максимизации награды.

В данной работе предлагается использование разновидности обучения с подкреплением, называемой Q-обучением. В классическом Q-обучении, в первую очередь необходимо определить пространство состояний и пространство действий. Робот исследует эти пространства и получает награду за каждое возможное действие. На каждом шаге, из каждого состояния агента, каждое действие получает награду, и награда для каждой пары состояние-действие сохраняется. Будущие действия предпринимаются на основании уже совершенных и преследуют цель максимизации текущей и будущей награды. В конце процедуры обучения, должна быть изучена последовательность действий при различных состояниях для достижения цели – походки по прямой линии с минимизацией крена и других нежелательных перемещений.

Для того, чтобы обучить робота походке при разных условиях, необходимо определить параметры, которые будут характеризовать как состояние окружающей среды (ландшафта), так и внутреннее состояние робота. В нашем случае основным источником информации об окружающей среде является зрение, а именно, RGBD-сенсор Microsoft Kinect. Данный сенсор предоставляет

информацию как о внешнем облике ландшафта (RGB-кадр), так и о его геометрической структуре (D-кадр). Что касается внутреннего состояния робота, оно складывается из состояний отдельных конечностей. Каждая конечность может пребывать в одном из трех состояний. Первое состояние – конечность поднята над землей. Два других – конечность спереди и сзади изначальной позиции (рис. 6):

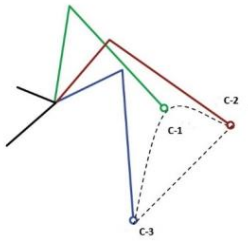

Рисунок 8 - Возможные состояния конечностей робота

Структура модели для обработки цветного изображения, получаемого из RGB-кадра сенсора Kinect, представлена на рисунке 9.

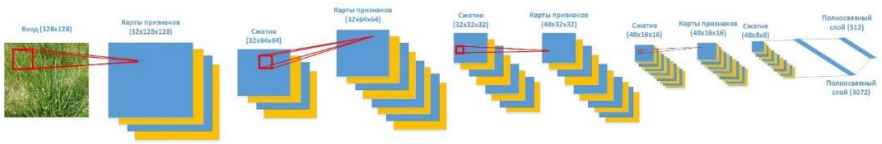

Рисунок 9 - Структура свёрточной нейронной сети для обработки цветного изображения

Для изучения высокоуровневых признаков ландшафта из цветного изображения используется свёрточная нейронная сеть. Основным структурным элементом подобных типов моделей является свёрточный (*convolutional*) слой, который применяет операцию свёртки к своему входу, получая на выходе так называемые *карты признаков*. За свёрточным слоем следует нелинейность,

применяемая к картам признаков. В свёрточных моделях широко используется нелинейность типа *rectified linear unit* (значения меньше нуля приравниваются к нулю). Она же используется и в данной работе. Опциональным элементом свёрточной сети является сжимающий (*pooling*) слой, производящий уменьшение изображения в размере путем усреднения или выбора пикселя максимальной интенсивности в определенной окрестности, установленной исходя из размера сжатия. Обычно в свёрточных сетях используется от двух до нескольких десятков комбинаций свёрточный слой – нелинейность – сжимающий слой, следующих друг за другом. На выходе сети присутствуют один или несколько полносвязных слоёв, которые позволяют агрегировать изученные признаки. Выходы последнего полносвязного слоя могут быть использованы для решения задачи классификации либо для дальнейшей обработки. В нашем случае значения выходов становятся частью описания общего состояния системы «робот и окружающая среда».

Используемая нами свёрточная сеть состоит из трех комбинаций свёрточный слой – нелинейность – сжимающий слой, за которыми следуют два полносвязных слоя. Мы используем свёрточный слой типа «same» (одинаковый). Это значит, что после свёртки размер изображения остается прежним. Сжатие происходит в окрестности 2x2 путём выбора пикселя максимальной интенсивности. Таким образом, каждый раз размер изображения уменьшается в 2 раза как по обоим измерениям.

Аналогичная структура применяется и для обработки D-канала. Мы используем специальную предобработку кадра глубины, трансформируя его в RGB-изображение. Значению дистанции ставится в соответствие комбинация значений RGB, где красный соответствует самой далёкой точке, а синий – самой ближней. Стоит отметить, что перед трансформацией D-кадр проходит фильтрацию градиентным фильтром [8] для закрашивания областей, недоступных для измерения и медианным фильтром для устранения небольших шумов. Схема сети для обработки D-канала представлена на рисунке 10:

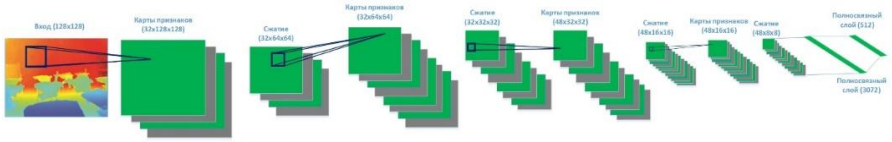

Рисунок 10 - Структура свёрточной нейронной сети для обработки кадра глубины

После прохождения данных по всей глубине обеих сетей, нам необходимо объединить изученные признаки (значения последнего полносвязного слоя) друг с другом, а также добавить значения внутреннего состояния робота. Для этого мы объединяем 3 вектора «по вертикали».

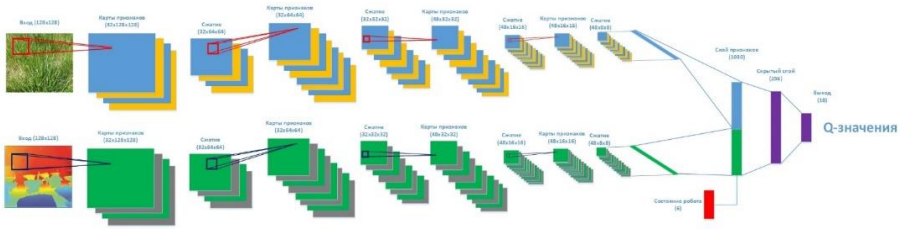

Рисунок 11 - Общая структура сети

Как видно из рисунка 9, входами каждой из компонент сети служат данные о состоянии робота и окружающей среды. Каждая часть данных проходит свои преобразования, а на стадии сложения признаков полученные векторы объединяются. Затем следует еще один скрытый слой, за которым следует выходной. Выходами являются так называемые Q-значения, то есть значения полезности каждого из возможных действий (переходов между состояниями). Этот момент будет подробно описан в следующей части статьи.

3.4.1. Алгоритм глубокого обучения с подкреплением

Обучение с подкреплением отличается от обучения с учителем тем, что в процессе обучения агенту даются не правильные ответы, но награды, которые позволяют ему понять, насколько хорошо он справляется с поставленной задачей. Поэтому целью обучающего алгоритма становится нахождение стратегии поведения, которая ведет к максимизации награды. Еще одной особенностью обучения с подкреплением является то, что обучение происходит в режиме реального времени, в процессе взаимодействия агента со средой.

Рисунок 12 - Схема обучения с подкреплением

$$\max_{\theta} \ \mathrm{E}[\sum_{t=0}^{H} R(s_t)|\pi_\theta]$$

Чтобы описать модель обучения, необходимо для начала определить наборы состояний и действий. Робот исследует пространство состояние-действие и получает награду за каждое возможное действие. Будущие действия предпринимаются в зависимости от уже совершенных и от значений полученных наград.

Как было описано ранее, состояние робота представляет собой набор состояний отдельных конечностей. Каждая конечность может пребывать в одном из трёх состояний, следовательно, состояние робота описывается шестью параметрами. Набор действий состоит из возможных переходов между

состояниями для каждой из конечностей (всего 36 действий, по 6 для каждой конечности). Конечно, некоторые из этих действий маловероятны. Например, когда конечность в состоянии 1 (поднята), возврат обратно в состояние 3 нелогично, или, если конечность в состоянии 2, возврат в 1 не поможет роботу при ходьбе, но, тем не менее, мы включаем эти действия в список возможных. Вполне возможно, что при обучении ходьбе на неровном ландшафте, какие-либо из этих действий могут оказаться полезными.

Для того, чтобы знать углы поворота сервоприводов, необходимые для переходов между состояниями, нужно решить обратную задачу кинематики. Так как в нашей задаче пространство состояний дискретно и конечно, мы предлагаем получить углы заранее для всех возможных состояний робота. Таким образом, мы получим таблицу, где каждому переходу между состояниями соответствуют углы, на которые необходимо повернуться сервоприводам. Эта таблица будет в дальнейшем использоваться для подачи непосредственных сигналов на моторы.

Одной из важнейших задач обучения с подкреплением является определение функции награды. Если полагать, что критерий того, насколько хорошо робот справляется с задачей ходьбы – это пройденное расстояние с одновременной минимизацией угла крена, то наиболее очевидным решением становится использование следующей формулы:

$$R(s,a) = \frac{движение\_вперед}{\sqrt{перемещение^2 + крен^2}}$$

Крен робота измеряется с помощью инерциального измерительного устройства. Изменения линейных координат фиксируются системой визуальной одометрии [13], использующей данные сенсора Kinect.

Перейдем непосредственно к алгоритму обучения нейронной сети. В данной работе мы используем вид обучения с подкреплением под названием Q-обучение. В Q-обучении мы определяем функцию *Q(s,a)*, которая представляет

собой максимальную отложенную награду при условии выполнения действия *a* в состоянии *s*:

$$Q(s_t, a_t) = \max R_{t+1}$$

Самый удобный способ представить себе смысл функции *Q(s,a)* – это «самый удачный счёт в конце игры после выполнения действия *a* в состоянии *s*». Под игрой в данном случае можно понимать большое разнообразие возможных задач, стоящих перед агентом. Эта функция называется Q-функцией, так как она представляет собой «качество» (Quality) определённого действия при данном состоянии.

Зная Q-значение для каждого действия, мы можем определить стратегию поведения агента:

$$\pi(\mathrm{s}) = \mathrm{argmax}_a \, Q(s, a)$$

Эта формула соответствует так называемому «жадному» алгоритму выбора действий. Однако, с этим уравнением связаны некоторые проблемы. Здесь стоит отметить, что одной из основных задач обучения является нахождение баланса между исследованием и стабильностью (exploration and exploitation problem). Суть этой проблемы заключается в том, что агент, найдя приемлемую стратегию действий, может остановиться на ней, приняв за наиболее удачную, и перестать предпринимать новые действия, значительно расходящиеся с текущей стратегией. Это может привести к тому, что агент упустит возможность найти более удачную стратегию поведения. Для того, чтобы бороться с этой проблемой, применяют алгоритм ε- жадного обучения (ε- greedy). При таком алгоритме, с вероятностью ε может быть выбрано случайное действие, а с вероятностью 1 - ε выбирается действие с наибольшим Q-значением. При реализации ε- жадного алгоритма может возникнуть следующая ситуация: в конце обучения, когда агент уже нашел лучшую стратегию поведения и Q-значения практически сошлись к оптимальным, выбирается случайное действие либо последовательность случайных действий, что может привести к дестабилизации и расхождению обучения. С целью борьбы с подобными

ситуациями, значение ε не фиксировано в процессе обучения, но уменьшается с течением времени. В таком случае, ближе к концу обучения, вероятность выбора случайного действия будет сведена к минимуму.

Теперь покажем, как мы получим Q-значения. Представим только один переход $< s, a, r, s' >$, где s – текущее состояние, a – предпринятое действие, r – полученная награда, s' – состояние, в которое пришел агент, когда предпринял действие a.

$$Q(s,a) = r + \gamma \max_{a'} Q(s',a')$$

Это уравнение называется уравнением Беллмана. Оно достаточно логично – максимальная отложенная награда для данного состояния и действия равна немедленной награде плюс максимальной отложенной награде для следующего состояния. Параметр γ показывает, насколько сильно мы принимаем во внимание будущие награды по сравнению с немедленной. Главная идея Q-обучения состоит в том, что мы можем итеративно аппроксимировать Q-функцию, используя уравнение Беллмана. В самом простом случае, Q-функция представлена в виде таблицы, с состояниями в строках и действиями в столбцах. Алгоритм Q-обучения представлен ниже:

Алгоритм 1. Классическое Q-обучение.

Инициализация Q[N_состояний, N_действий] случайными значениями

Получение данных о начальном состоянии ***s***

Повторение

Выбор и исполнение действия ***a***

Получение данных о награде **r** и о новом состоянии ***s'***

$Q(s,a) = Q(s,a) + \alpha(r + \gamma \max_{a'} Q(s',a') - Q(s,a))$

s = ***s'***

до прекращения эпизода

Под эпизодом в данном случае понимается последовательность переходов от начального состояния до терминального. За терминальное состояние может

быть принято столкновение робота с препятствием, потеря стабильности или отсутствие прогресса в передвижении в течение определенного времени. α в данном алгоритме – параметр обучения, который контролирует, насколько сильно разница между текущим и предложенным Q-значением принимается во внимание. Если $\alpha = 1$, то два слагаемых $Q(s,a)$ взаимоуничтожаются и мы получаем уравнение Беллмана.

Если бы мы попытались представить Q-функцию в виде таблицы в нашей задаче, мы бы столкнулись с так называемой проблемой размерности. Если принимать во внимание только состояние робота, которое в данном случае можно представить, как совокупность состояний конечностей, которые описываются дискретными переменными, то можно прийти к конечному количеству состояний и действий и составить Q-таблицу. Но имея на входе визуальные данные, мы получаем бесконечное количество возможных состояний, и, принимая этот факт во внимание, нам необходимо использовать другое представление Q-функции. В данном случае, мы используем глубокую нейронную сеть.

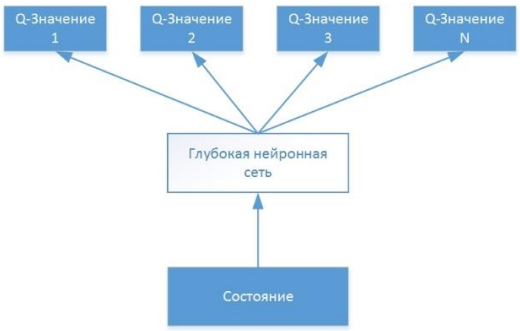

Рисунок 13 - Аппроксимация Q-функции в виде глубокой нейронной сети

Q-функция может принимать любые значения, что означает обучение нейронной сети для решения задачи регрессии. Оптимизационная задача может быть решена с помощью следующей функции потерь:

$$L = \frac{1}{2}[r + \gamma \max_{a'} Q(s', a') - Q(s, a)]^2$$

Имея переход $< s, a, r, s' >$, алгоритм обучения примет следующий вид:

Алгоритм 2. Глубокое Q-обучение.

1. Прямой проход через сеть для текущего состояния *s* для получения ожидаемых Q-значений для действий.
2. Выбор действия с максимальным Q-значением, либо в соответствии с ε - жадным алгоритмом.
3. Получение информации о награде *r*
4. Вычисление или симуляция перехода из *s* в следующее состояние *s'*.
5. Прямой проход через сеть для следующего состояния *s'* и вычисление значения $\max_{a'} Q(s', a')$.
6. Вычисление целевого значения $r + \gamma \max_{a'} Q(s', a')$ для действия, выбранного на шаге 2. Для остальных действий задать целевое Q-значение равным оригинальному, полученному на шаге 1, полагая ошибку для соответствующих выходов равной нулю.
7. Обновление весов нейронной сети методом обратного распространения ошибки.

3.4.2. Глубокое обучение с подкреплением в непрерывном пространстве действий.

Как можно заметить, описанный в предыдущих разделах алгоритм глубокого Q-обучения работает в непрерывном пространстве состояний, но в дискретном пространстве действий. Каждая конечность робота-гексапода может находиться в одном из трех дискретных состояний, соответственно робот может находиться в одном из 3^6 = 729 состояний. Набор действий формируется из

всевозможных переходов между состояниями. Такой подход имеет свои плюсы и минусы. К плюсам можно отнести относительную простоту реализации Q-обучения. Из существенных минусов стоит выделить невозможность исследования таких действий, как поднимание-опускание корпуса, увеличение ширины шага, движение боком, т.е. всех действий, выполнение которых требует нахождения робота в состояниях, отличных от заданных нами.

Более перспективным вариантом стала бы модель обучения, которая предполагает переход от входных данных состояния непосредственно к углам поворота сервомоторов. Тогда робот мог бы, не имея начальных знаний о возможных действиях, без привлечения обратной задачи кинематики, находить стратегии передвижения. К сожалению, Q-обучение в классическом его варианте, в том числе и глубокое Q-обучение, не подходят для решения подобной задачи, так как сама цель Q-обучения – предоставление значений полезности для конечного числа действий.

Для осуществления обучения с подкреплением в непрерывном пространстве действий (непрерывного контроля), в основном используется архитектура Актер – Критик. Данная архитектура предполагает разделение вычисления Q-значений и выбора действий. Представленный одной глубокой сетью, актер выдает на выходе непрерывное действие, а представленный второй глубокой сетью критик оценивает *Q*-значение для данного действия. Сеть актера μ, параметризованная значениями θ^{μ}, принимает на вход состояние *s* и выдает непрерывное действие *a* (в нашем случае, набор углов сервоприводов). Сеть критика *Q*, параметризованная значениями θ^{Q}, принимает на вход состояние *s* и действие a, и выдает на выходе скалярное *Q*-значение *Q(s,a).*

Метод изменения весов сети критика остаётся неизменным по сравнению с тем, что был представлен в предыдущем разделе:

$$Q(s,a) = Q(s,a) + \alpha(r + \gamma \max_{a'} Q(s',a') - Q(s,a))$$

Адаптируя к случаю нейронной сети:

$$L_Q(s,a \mid \theta^Q) = (Q(s,a \mid \theta^Q) - (r + \gamma \max Q(s',a' \mid \theta^Q)))^2$$

Однако, для непрерывного состояния действий это уравнение не подходит, так как включает максимизацию *Q*-значений для следующего состояния. Вместо этого, мы просим актера предоставить нам следующее действие $a' = \mu(s', \theta^{\mu})$. Это приводит к следующей функции потерь для критика:

$$L_Q(s, a \mid \theta^Q) = (Q(s, a \mid \theta^Q) - (r + \gamma Q(s', \mu(s' \mid \theta^{\mu}) \mid \theta^Q)))^2$$

Аппроксимация Q-функции критика может быть получена путем оптимизации методом градиентного спуска для представленной функции потерь по отношению к параметрам θ^{μ}. Однако, адекватность этой аппроксимации существенно зависит от качества стратегии, выбранной актёром, потому что именно актёр определяет следующее действие *a'*, использующееся для обновления весов критика.

Знания критика о Q-значениях действий можно использовать для обучения стратегии актёра. Имея определенное состояние, цель актера – минимизировать разницу между текущим выбранным действием *a* и оптимальным действием *a** в этом состоянии:

$$L_{\mu}(s \mid \theta^{\mu}) = (a - a^*)^2 = (\mu(s \mid \theta^Q) - a^*)^2$$

Критик может быть использован для предоставления информации о качестве разных действий, но прямое оценивание *a** будет подразумевать максимизацию выхода критика по всем возможным действиям:

$$a^* \approx \operatorname{argmax}_a Q(s, a \mid \theta^Q)$$

Вместо подобного поиска глобального максимума, сеть критика может предоставить градиенты, которые укажут направления изменения (в пространстве действий), которые приведут к большим Q-значениям: $\nabla_a Q(s, a \mid \theta^Q)$. Получить эти градиенты можно, совершив один обратный проход по сети критика, что намного быстрее, чем решение оптимизационной задачи в непрерывном пространстве действий. Заметим, что эти градиенты функции потерь получены не по отношению к параметрам θ^Q, но по отношению ко входам. Для того, чтобы обновить веса актера, необходимо поместить эти

градиенты на место значений выходного слоя актера и совершить обратный проход по сети.

3.5. Объединение разработанных модулей в систему навигации

В предыдущем разделе были описаны алгоритмы, которые используются роботом-гексаподом для решения задач походки по прямой линии, локализации и построения карты местности. Для того, чтобы объединить эти модули в систему навигации, необходимо разработать алгоритм поиска и планирования пути. После проведенного исследования, было решено использовать алгоритм поиска пути под названием A*, который является модификацией классического поиска в ширину, но при этом использует эвристическую функцию. Подробнее основные шаги алгоритма описаны ниже.

1. Создадим массив структур типа node (узел) по размеру карты, получаемой из модуля построения карты

```
struct node
{
    float x;            // координаты узла в сетке
    float y;
    int is_occupied;    // занят или свободен узел
    int is_visited;     // посещен ли узел
    int g;              // значение g, необходимое для поиска пути
    int when_expanded;  // на каком шаге посещен узел
    int action_to_get;  // откуда пришли в узел
}
```

2. Создадим пустой символьный массив по размеру карты. Заполним его тогда, когда будем простраивать путь от конца к началу.
3. Создадим массив стоимостей. Каждый элемент массива – декартово расстояние от точки на карте, соответствующей данному элементу, до цели. Использование этого массива – отличительная особенность A*.
4. Начальный узел помещается в открытый список, и помечается как посещённый (node.is_visited = 1)

5. Открытый список сортируется по значению g, которое складывается из стоимости пути из начальной точки в текущую плюс стоимость пути из текущей точки в целевую (узел с наименьшим значением – в конце списка). Следующим узлом назначается последний узел списка. Он же удаляется из списка.
6. Если координаты текущего узла равны координатам цели, заканчиваем поиск.
7. Осматриваются все соседи текущего узла. В данной реализации осматриваются только 4 соседа (левый, правый, верхний, нижний). Если координаты узла не выходят за пределы карты, узел не посещен и не занят, то он помещается в открытый список.
8. Если список пуст, заканчиваем поиск, пути нет.
9. Если список не пуст, на шаг 5.

После того, как путь найден, строим его от конца к началу благодаря параметру node.action_to_get.

Сглаживание пути

Даже принимая во внимание тот факт, что построенный путь весьма высокого разрешения, применим к нему алгоритм сглаживания:

1. Создадим новый путь размера найденного несглаженного пути и присвоим значениям координат его узлов значения координат узлов старого пути:

$$y_i = x_i;$$

2. Оптимизируем путь согласно двум критериям:

$$(x_i - y_i)^2 \to \min$$
$$(\mathrm{y}_i - y_{i+1})^2 \to \min$$

Реализуем данную оптимизацию с помощью метода градиентного спуска.

$$y_i = y_i + \alpha \cdot (x_i - y_i);$$
$$y_i = y_i + \beta \cdot (y_{i+1} + y_{i-1} + 2 \cdot y_i);$$

Будем итеративно производить данную операцию, и в конце каждой итерации вычислять абсолютное значение разницы значений координат на предыдущей итерации и текущей. Когда это значение будет меньше определенного порога (мы использовали 0,001), то это значит, что путь оптимизирован. Значения коэффициентов α и β подбираются экспериментально (в пределах от 0 до 1) в зависимости от того, какая степень сглаживания будет оптимальной для данной карты. Если требуется очень высокая точность передвижения (среда с множеством препятствий), значения коэффициентов должны стремиться к нижнему пределу.

Все части системы навигации объединяются в один конвейер управления с помощью операционной системы Robot Operating System (ROS). ROS представляет собой набор библиотек и функций для решения различных задач робототехники, а также предоставляет средства для взаимодействия между различными программными узлами, которые выполняют различные задачи. В данном случае, в системе навигации задействованы следующие узлы:

1) Узел сбора данных с сенсора Kinect. Использует API OpenNI2 для доступа к данным сенсора. Публикует сообщения типа /camera/depth_frame, а также сообщение типа /camera/rgb_frame
2) Узел фильтрации данных с сенсора Kinect. Применяет градиентный и медианный фильтры к оригинальному кадру глубины. Публикует сообщение типа /camera/depth_frame
3) Узел приема данных с IMU. Публикует сообщение с полями ориентации вокруг осей *x,y,z* в топик с названием **/imu/data**
4) Узел визуальной одометрии. Подписан на топик с сообщениями от сенсора Kinect. Публикует сообщение с линейными и угловыми перемещениями робота.

5) Узел симуляции лазерного сканера. Подписан на топик с данными глубины от сенсора Kinect. Публикует сообщение типа LaserScan.
6) Узел локализации и построения карты. Подписан на топик, в который публикует данные узел визуальной одометрии, а также на топик с данными от симулятора лазерного сканера. Публикует два сообщения: изображение карты и глобальную позицию робота.
7) Узел генерации управляющих сигналов для двигателей. Получает данные от узлов сенсора Kinect и инерциального измерительного устройства. Представляет собой нейронную сеть, структура которой описана в разделе 3.4.
8) Узел передачи данных по последовательному порту на микроконтроллер. Использует инструментарий библиотеки Qt, модуль QtSerialPort.
9) Узел поиска пути. Использует алгоритм A* для поиска кратчайшего пути до цели в соответствии с текущей картой. Подписан на топик с данными карты и глобальной позицией робота. Публикует расстояние до глобальной и локальных целей, а также угле необходимого поворота для того, чтобы идти к локальной цели по прямой линии, используя алгоритм походки.

Технико-экономическое обоснование НИР

Введение

В настоящее время мобильные роботы используются для выполнения множества разнообразных задач. Их список включает в себя обслуживание складских помещений, перевозку грузов, исследование и мониторинг труднодоступных или опасных зон, разведку, взятие проб. Для этих целей было создан и исследован широкий спектр конструкций роботов. Последнее десятилетие во многих областях все большее применение находят шагающие роботы. Они обладают рядом очевидных преимуществ перед своими колесными или гусеничными аналогами. Шагающие роботы могут двигаться на неровном ландшафте и проникать в труднодоступные места благодаря возможности гибкого выбора модели передвижения. Также они могут сохранять дееспособность при потере одной или нескольких конечностей, перестраивая модель передвижения на ходу (при наличии соответствующей конструкции и алгоритмов управления). Целью проведения научно-исследовательской работы является разработка системы управления для робота-гексапода, которая будет выполнять три основные задачи: передвижение по неровному ландшафту, распознавание объектов, построение карты местности.

Цель раздела – комплексное описание и анализ финансово-экономических аспектов выполненной работы. Необходимо оценить полные денежные затраты на исследование (проект), а также дать экономическую оценку результатов ее внедрения.

4.1. Затраты по основной заработной плате исполнителей темы

Определение трудоемкости выполнения работ основными исполнителями темы.

Процесс разработки делится на три этапа:

- подготовительный;
- основной;
- заключительный.

Продолжительность работ ($t_{ож}$) определяется либо по нормативам (с использованием специальных справочников) для каждого исполнителя в отдельности, либо расчетом с помощью экспертных оценок по формуле:

$$t_{ожс} = \frac{3t_{\min} + 2t_{\max}}{5}, \quad (1)$$

где t_{min} – минимальная трудоемкость работ, ч.-дн.;

t_{max} –максимальная трудоемкость работ, ч.-дн.

Сроки t_{min} и t_{max} устанавливает руководитель.

Для расчета заработной платы основных исполнителей проекта необходимо ожидаемое время перевести в рабочее, для этого нужно:

$$T_{раб} = t_{ож} \cdot К_{д}, \quad (2)$$

где $К_{Д}$ – коэффициент, учитывающий дополнительное время на компенсации и согласование работ ($К_{Д}$=1.2).

Для удобства построения календарного план-графика, длительность этапов в рабочих днях переводится в календарные дни и рассчитывается по следующей формуле:

$$t_{кд} = t_{раб} \times t_{к}, \text{ кал. Дн.}, \quad (3)$$

где: $t_{КД}$ – продолжительность выполнения этапа в календарных днях;

$t_{К}$ – коэффициент календарности.

Коэффициент календарности рассчитывается по формуле:

$$t_{к} = \frac{t_{кг}}{t_{кг} - t_{вд} - t_{пд}}, \quad (4)$$

где: $t_{кг}$ – количество календарных дней в году;

$t_{вд}$ – количество выходных дней в году;

$t_{пд}$ – количество праздничных дней в году.

$$t_{к} = \frac{365}{365 - 65} = 1.22$$

Результаты расчетов представлены в таблице 4.

Этапы работы представлены в таблице 3

Таблица 3. Этапы работы над проектом

Этапы работы	Исполнители	Загрузка исполнителей
Постановка целей и задач, получение исходных данных	НР	НР — 100%
Составление и утверждение ТЗ	НР, И	НР — 100% И — 40%
Изучение литературы	И	И — 100%
Подготовка теоретической базы исследований	НР, И	НР — 40% И — 100%
Разработка и тестирование программного обеспечения	И	И — 100%
Анализ полученных данных	НР, И	НР — 40% И — 100%
Оформление расчетно-пояснительной записки	И	И — 100%
Подведение итогов	НР, И	НР — 50% И — 100%

Таблица 4. Трудоемкость выполнения работ в проекте

Этап	Исполнители	Продолжительность работы, дни			Трудоемкость работ по исполнителям чел.-дн.			
		$t_{\min}$	$t_{\max}$	$t_{ож}$	$T_{РД}$		$T_{КД}$	
					НР	И	НР	И
1	2	3	4	5	6	7	8	9
Постановка целей и задач, получение исходных данных	НР	3	7	4.6	5.5	0	8.2	0
Составление и утверждение ТЗ	НР, И	15	30	21	25.2	10.1	37.2	14.9
Изучение литературы	И	10	14	11.6	0	13.9	0	20.6
Подготовка теоретической базы исследований	НР, И	40	45	42	25.2	50.4	37.2	74.5
Разработка и тестирование программного обеспечения	И	35	45	39	0	46.8	0	69.2
Анализ полученных данных	НР, И	40	45	42	0	50.4	0	74.5
Оформление расчетно-пояснительной записки	И	30	40	34	0	40.8	0	60.3
Подведение итогов	НР, И	2	2	2	1.7	2.4	2.5	3.5
Итого				196.2	57.6	214.8	85.1	317.4

Удельное значение каждого этапа в процентах определяется по формуле:

$$I_i = \frac{t_{ожi}}{t_{ож}} \cdot 100\%, \quad (5)$$

где I_i – удельное значение каждого этапа в %;

$t_{ожi}$ – трудоемкость этапа;

$t_{ож}$ – суммарная трудоемкость.

Нарастание технической готовности рассчитывается по формуле:

$$H_i = \frac{t_н}{t_{ож}} \cdot 100\%, \quad (6)$$

где H_i – нарастание технической готовности i – го этапа в %;

$t_н$ – нарастающая трудоемкость с начала работы i – го этапа.

При построении линейного графика должен быть охвачен весь спектр работ по теме. Учитывая тот факт, что дипломник выступает в качестве основного исполнителя, занятого выполнением темы на протяжении всего периода её проведения, на линейном графике не должно быть перерывов в работе дипломника. Для проведения отдельных видов работ необходимо привлекать руководителя проекта, на графике это будет показано в виде параллельных линий, характеризующих одновременное проведение работ нескольких видов работ разными исполнителями.

Линейный график выполнения работ и выполненные расчеты приведены в таблице 5.

Таблица 5. Календарный план-график работ

Наименование работы	I_i **,%**	H_i **,%**	t_{ki} **кал дн.**	**Продолжительность выполнения работ, дни**											
				I квартал			**II квартал**			**III квартал**			**IV квартал**		
				30	60	90	120	150	180	210	240	270	300	330	360
Постановка целей и задач, получение исходных данных	2,0	2,0													
			8,2												
Составление и утверждение ТЗ	13,0	15,0	14,9												
			37,2												
Изучение литературы	5,1	20,1	20,6												
Подготовка теоретической базы исследований	27,8	47,8	74,5												
			37,2												
Разработка и тестирование программного обеспечения	17,2	65,0	69,2												
Анализ полученных данных	18,5	83,5	74,5												
Оформление расчетно-пояснительной записки	15,0	98,5	60,3												
Подведение итогов	1,5	100	3,5												
			2,5												

– инженер

– научный руководитель

4.2. Расчет затрат на материалы

К данной статье расходов относится стоимость материалов, покупных изделий, полуфабрикатов и других материальных ценностей, расходуемых непосредственно в процессе выполнения работ над объектом проектирования. Сюда же относятся специально приобретенное оборудование, инструменты и прочие объекты, относимые к основным средствам, стоимостью до 40 000 руб. включительно. Цена материальных ресурсов определяется по соответствующим ценникам или договорам поставки. Кроме того статья включает так называемые транспортно-заготовительные расходы, связанные с транспортировкой от поставщика к потребителю, хранением и прочими процессами, обеспечивающими движение (доставку) материальных ресурсов от поставщиков к потребителю. Сюда же включаются расходы на совершение сделки купли-продажи (т.н. транзакции). Приближенно они оцениваются в процентах к отпускной цене закупаемых материалов, как правило, это 5-20 %.

Таблица 6. Расчет затрат на материалы

Наименование материалов	Цена за ед., руб.	Кол-во	Сумма, руб.
Сенсор Microsoft Kinect	4000	1	4000
Компьютер Jetson TK1	12000	1	12000
Комплект контроллера Arduino Mega	5600	1	5600
Стереокамера RGB	2500	1	2500
Итого:			**24100**

4.3. Расчет заработной платы основных исполнителей проекта

Затраты на оплату труда планируются с учетом продолжительности выполнения темы и ее отдельных этапов, степени занятости исполнителей темы (для некоторых категорий – трудоемкости работ), с использованием данных о нормах оплаты их труда. Для расчета заработной платы по штатно-окладной системе оплаты труда используется формула

$$ЗП_{осн} = \sum_{i=1}^{n} t_{рабі} \cdot ОК_{мес} \cdot Ч_{испі}\ , \quad (7)$$

где $t_{рабі}$ – продолжительность выполнения работы, раб. Дн.;

$ОК_{мес}$ – месячные оклады исполнителей, руб./ мес.;

$Ч_{испі}$ – количество исполнителей i-ой категории.

Размер основной заработной платы устанавливается, исходя из численности исполнителей, трудоемкости и средней заработной платы за один рабочий день.

$$ЗП_{осн} = \sum_{i=1}^{n} T_i \cdot СЗП\ , \quad (8)$$

где n - количество участников в i-ой работе,

T_i - затраты труда (трудоемкость), необходимые для выполнения i-го вида работ, (дни). Трудоемкость определяется по таблице 2, находится количество дней, которое необходимо потратить на разработку системы.

СЗП - среднедневная заработная плата исполнителя, выполняющего i-ый вид работ, (руб./дней). Среднедневная заработная плата рассчитывается следующим образом:

$$СЗП = \frac{\text{месячный оклад}}{\text{количество рабочих дней}}, \quad (9)$$

Количество рабочих дней в месяце примем равным 24,83.

Расчеты затрат на полную заработную плату приведены в таблице 5 Затраты времени по каждому исполнителю в рабочих днях с округлением до целого взяты из таблицы 2. Для учета в ее составе премий, дополнительной зарплаты и районной надбавки используется следующий ряд коэффициентов: $К_{ПР}$ = 1,1; $К_{доп.ЗП}$ = 1,188; $К_р$ = 1,3. Таким образом, для перехода от тарифной (базовой) суммы заработка исполнителя, связанной с участием в проекте, к

соответствующему полному заработку (зарплатной части сметы) необходимо первую умножить на интегральный коэффициент $К_и$ = 1,1* 1,188*1,3 = 1,699. Вышеуказанное значение $К_{доп.ЗП}$ применяется при шестидневной рабочей неделе, при пятидневной оно равно 1,113, соответственно в этом случае $К_и$ = 1,62.

Таблица 7. Затраты на основную заработную плату

Исполнитель	**Оклад, руб./раб. день**	**Среднедневная ставка, руб./раб. день**	**Затраты времени, раб. дни**	**Коэффициент**	**Фонд з/платы, руб.**
НР	32162.9	1295.3	66	1.699	145249.9
И	7483.6	301.4	245	1.699	125456.5
Итого:					**270706.4**

4.4. Расчет затрат на социальный налог

Затраты на единый социальный налог (ЕСН), включающий в себя отчисления в пенсионный фонд, на социальное и медицинское страхование, составляют 30 % от полной заработной платы по проекту, т.е. $C_{\text{СОЦ.}} = C_{ЗП} \cdot 0,3$. В нашем случаем получим: $C_{\text{СОЦ.}} = 270706,4 \cdot 0,3 = 81211,9$

4.5. Расчет затрат на электроэнергию

Данный вид расходов включает в себя затраты на электроэнергию, потраченную в ходе выполнения проекта на работу используемого оборудования, рассчитываемые по формуле:

$$С_{эл.об.}=Р_{об} \cdot t_{об} \cdot Ц_Э \quad (10)$$

где $P_{ОБ}$ – мощность, потребляемая оборудованием, кВт;

$Ц_Э$ – тариф на 1 кВт·час;

$t_{об}$ – время работы оборудования, час.

Для ТПУ $Ц_Э$ = 5,257 руб./квт·час (с НДС).

Время работы оборудования вычисляется на основе итоговых данных таблицы 2 для инженера ($T_{РД}$) из расчета, что продолжительность рабочего дня равна 8 часов.

$$t_{об} = T_{РД} * К_t, \quad (11)$$

где $К_t \leq 1$ – коэффициент использования оборудования по времени, равный отношению времени его работы в процессе выполнения проекта к $T_{РД}$, определяется исполнителем самостоятельно. В ряде случаев возможно определение $t_{об}$ путем прямого учета, особенно при ограниченном использовании соответствующего оборудования.

Мощность, потребляемая оборудованием, определяется по формуле:

$$P_{ОБ} = P_{ном.} * К_С \quad (12)$$

где $P_{ном.}$ – номинальная мощность оборудования, кВт;

$K_С \leq 1$ – коэффициент загрузки, зависящий от средней степени использования номинальной мощности. Для технологического оборудования малой мощности $K_С = 1$.

Пример расчета затраты на электроэнергию для технологических целей приведен в таблице 8.

Таблица 8. Затраты на электроэнергию

Наименование оборудования	Время работы оборудования $t_{ОБ}$, час	Потребляемая мощность $P_{ОБ}$, кВт	Затраты $Э_{ОБ}$, руб.
Персональный компьютер	1720*1	0,4	3616,8
Итого:			**3616,8**

4.6. Амортизация основных фондов

*Данный элемент о*тражает сумму амортизационных отчислений (части, перенесённые по мере физического износа основных средств на производимый продукт) на полное восстановление основных средств используемых при реализации проекта. К амортизируемым основным фондам относится оборудование, стоимость которого выше 20000 рублей и срок

эксплуатации более года. В противном случае оно включается в материальные расходы.

Амортизационные отчисления рассчитываются по формуле:

$$C_{АМ} = \frac{H_А \cdot Ц_{ОБ} \cdot t_{рф} \cdot n}{F_Д}, \quad (13)$$

где $Н_А$ – годовая норма амортизации единицы оборудования;

$Ц_{ОБ}$ – балансовая стоимость единицы оборудования с учетом ТЗР. При невозможности получить соответствующие данные из бухгалтерии она может быть заменена действующей ценой, содержащейся в ценниках, прейскурантах и т.п.;

$F_Д$ – действительный годовой фонд времени работы соответствующего оборудования, берется из специальных справочников или фактического режима его использования в текущем календарном году. При этом второй вариант позволяет получить более объективную оценку $С_{АМ}$.

$t_{рф}$ – фактическое время работы оборудования в ходе выполнения проекта, учитывается исполнителем проекта;

n – число задействованных однотипных единиц оборудования.

На - норма амортизационных отчислений (%), которая в соответствии с Налоговым кодексов РФ определяется по следующей формуле:

$$H_a = \frac{1}{Tп.и.} \cdot 100\%, \quad (14)$$

где $Т_{п.и.}$ – срок полезного использования объекта (в годах) определяется в соответствии с классификацией основных средств, включаемых в амортизационные группы.

В ходе выполнения работы не требовалось оборудование, для которого необходимы амортизационные отчисления.

4.7. Прочие расходы

Накладные расходы учитывают прочие затраты организации, не попавшие в предыдущие статьи расходов: оплата услуг связи, электроэнергии, почтовые расходы, размножение материалов и т.д. Их величина определяется по следующей формуле:

$$C_{проч} = C_{осн} \cdot k_{нр}, \qquad (15)$$

где $k_{нр}$ – коэффициент, учитывающий накладные расходы. Принимается равным 10 % от общей суммы расходов.

В нашем случае прочие расходы составят: $C_{проч} = 379633 \cdot 0,1 = 37963,3$

4.8. Расчет общей себестоимости разработки

Себестоимость разработки АСУ определяется суммой затрат:

- материальных затрат;
- затраты на основную заработную плату;
- отчисления во внебюджетные фонды;
- расходы на электроэнергию;
- амортизация основных фондов;
- прочие расходы.

Таблица 9. Себестоимость разработки системы

Наименование статьи	**Сума, руб.**
Материальные затраты, руб.	24100
Затраты на основную заработную плату, руб.	270706
Отчисления во внебюджетные фонды, руб.	81211.1
Расходы на электроэнергию, руб.	3 616.8
Амортизация основных фондов, руб.	0
Прочие расходы, руб.	37963,3
Итого, руб.	**417596**

Так как мы не располагаем информацией о прибыли, то примем ее равной 20% от расходов на разработку, т.о. получим $417596 \cdot 0.2 = 83519$

НДС составляет 18% от суммы затрат на разработку и прибыли. В нашем случае это: $(417596 + 83519) \cdot 0,18 = 90200,7$

Цена разработки равна сумме полной себестоимости, прибыли и НДС, в нашем случае: $417596 + 83519 + 90200 = 591315$

4.9. Оценка экономической эффективности разработанной системы

Проект направлен на разработку системы управления для робота-гексапода на основе глубокого машинного обучения с подкреплением. Результаты, полученные в ходе проекта, имеют научную ценность и на текущей стадии развития разработки не ориентированы на получение экономического эффекта.

4.10. Оценка научно-технического уровня НИР

Научно-технический уровень характеризует влияние проекта на уровень и динамику обеспечения научно-технического прогресса в данной области. Для оценки научной ценности, технической значимости и эффективности, планируемых и выполняемых НИР, используется метод балльных оценок. Балльная оценка заключается в том, что каждому фактору по принятой шкале присваивается определенное количество баллов. Обобщенную оценку проводят по сумме баллов по всем показателям. На ее основе делается вывод о целесообразности НИР.

Сущность метода заключается в том, что на основе оценок признаков работы определяется интегральный показатель (индекс) ее научно-технического уровня по формуле:

$$I_{НТУ} = \sum_{i=1}^{3} R_i \cdot n_i, \qquad (16)$$

где $I_{НТУ}$ – интегральный индекс научно-технического уровня;

R_i – весовой коэффициент i-го признака научно-технического эффекта;

n_i – количественная оценка i-го признака научно-технического эффекта, в баллах.

Таблица 10. Весовые коэффициенты признаков НТУ

Признаки научно-технического эффекта НИР	**Характеристика признака НИР**	**Ri**
Уровень новизны	Систематизируются и обобщаются сведения, определяются пути дальнейших исследований	0,4
Теоретический уровень	Разработка способа (алгоритм, программа мероприятий, устройство, вещество и т.п.)	0,1
Возможность реализации	Время реализации в течение первых лет	0,5

Таблица 10. Баллы для оценки уровня новизны

Уровень новизны	**Характеристика уровня новизны – n_1**	**Баллы**
Принципиально новая	Новое направление в науке и технике, новые факты и закономерности, новая теория, вещество, способ	8 – 10
Новая	По-новому объясняются те же факты, закономерности, новые понятия дополняют ранее полученные результаты	5 – 7
Относительно новая	Систематизируются, обобщаются имеющиеся сведения, новые связи между известными факторами	2 – 4
Не обладает новизной	Результат, который ранее был известен	0

Таблица 11. Баллы значимости теоретических уровней

Теоретический уровень полученных результатов – n_2	**Баллы**
Установка закона, разработка новой теории	10
Глубокая разработка проблемы, многоспектральный анализ взаимодействия между факторами с наличием объяснений	8
Разработка способа (алгоритм, программа и т. д.)	6

Элементарный анализ связей между фактами (наличие гипотезы, объяснения версии, практических рекомендаций)	2
Описание отдельных элементарных факторов, изложение наблюдений, опыта, результатов измерений	0,5

Таблица 12. Возможность реализации результатов по времени

Время реализации – n_3	**Баллы**
В течение первых лет	10
От 5 до 10 лет	4
Свыше 10 лет	2

Так как все частные признаки научно-технического уровня оцениваются по 10-балльной шкале, а сумма весов Ri равна единице, то величина интегрального показателя также принадлежит интервалу [0, 10].

В таблице 12 указано соответствие качественных уровней НИР значениям показателя, рассчитываемого по формуле (16).

Таблица 12. Уровни НТЭ

Уровень НТЭ	**Показатель НТЭ**
Низкий	1-4
Средний	4-7
Высокий	8-10

Для используемого в пособии примера частные оценки уровня n_i и их краткое обоснование даны в таблице 13.

Таблица 13. Оценки научно-технического уровня НИР

Значимость	Фактор НТУ	Уровень фактора	Балл	Обоснование выбранного балла
0,4	Уровень новизны	Новая	6	Новое направление в технике, называемое глубоким обучением, применено к управлению шагающим роботом
0,1	Теоретический уровень	Разработка способа	6	Разработан способ применения глубокого обучения с подкреплением для обучения шагающего робота походке на неровном ландшафте
0,5	Возможность реализации	В течение первых лет	10	Быстрое воспроизведение проведенных исследований

Отсюда интегральный показатель научно-технического уровня для нашего проекта составляет:

$$I_{\text{нту}} = 0,4*6 + 0,1*6 + 0,5*10 = 2,4 + 0,6 + 5 = 8$$

Таким образом, исходя из данных таблицы 13, данный проект имеет высокий уровень научно-технического эффекта.

В данном разделе выпускной квалификационной работы дается характеристика проводимым работам, рабочему месту и рабочей зоне. Проанализированы опасные и вредные факторы труда, а также разработан комплекс мероприятий, снижающий негативное воздействие проводимой деятельности на работников и окружающую среду.

Основная часть данной работы выполнялась в лаборатории робототехники ТПУ студенческого бизнес инкубатора. Выбор места проведения работ определялся наличием необходимого для выполнения поставленных задач оборудования.

Основным недостатком данной лаборатории является почти полное отсутствие естественного освещения. Освещение осуществляется лампами. Температура в помещении постоянна. Зимой включается центральное отопление.

В аудитории присутствуют рабочие столы, оснащенные персональными компьютерами. Мониторы расположены на уровне глаз, достаточно места для комфортной посадки и работы. Уровень шума в помещении находится на приемлемом уровне. Лаборатория оснащена компьютерами, каждый компьютер подключается к сети с помощью сетевого фильтра со встроенным предохранителем. Каждая розетка индивидуально выключается на электрическом щите. Все токоведущие провода спрятаны в кожухи, нет оголенных проводов. Электробезопасность полностью соблюдается.

В рамках дипломной работы разрабатывается система управления для автономного робота-гексапода. Программная часть разработки выполняется на персональном компьютере. Аппаратная часть проектировалась с использованием микросхем, сенсоров и других электронных компонентов, напряжение питания которых составляет не более 6В.

5.1. Анализ вредных производственных факторов

К вредным производственным факторам относят: производственные метеоусловия, вредные вещества, виброакустические поля, производственное освещение, электромагнитные излучения, ионизирующие излучения.

Так как ВКР имеет физико-техническую тематику, то анализ вредных факторов будет начинаться электромагнитного и ионизирующего излучения, затем необходимо рассмотреть влияние производственных метеоусловий, виброакустических полей и производственного освещения. Стоит учесть, что разработчик не подвергается воздействию каких-либо вредных веществ, следовательно, данный производственный фактор рассматриваться не будет.

5.1.1. Электромагнитное излучение

При работе, компьютер образует вокруг себя электромагнитное поле, которое деионизирует окружающую среду.

Согласно СанПиН 2.22.542-96 напряженность электромагнитного поля на расстоянии 50 см вокруг монитора по электрической составляющей должна быть не более:

— В диапазоне частот 5 Гц ÷ 2 кГц – 25 В/м;

— В диапазоне частот 2 кГц ÷ 400кГц – 2,5 В/м.

Плотность магнитного потока должна быть не более:

— В диапазоне частот 5 Гц ÷ 2 кГц – 250 нТл;

— В диапазоне частот 2 кГц ÷ 400кГц – 25 нТл.

Возможные способы защиты от электромагнитного излучения:

1. По возможности, стоит использовать жидкокристаллический монитор, поскольку его излучение значительно меньше, чем у распространённых ЭЛТ мониторов (монитор с электроннолучевой трубкой). В данной работе использовался жидкокристаллический монитор BenQ gw2760.

2. Системный блок и монитор должны находиться как можно дальше от разработчика.

3. В связи с тем, что электромагнитное излучение от стенок монитора намного больше, необходимо устанавливать мониторы корпусом к стене, чтобы излучение поглощалось стеной.

4. По возможности сократить время работы за компьютером и делать частые перерывы.

5. Компьютер должен быть заземлён. Если имеется защитный экран, то он также должен быть заземлен.

5.1.2. Микроклимат помещения

Большое внимание необходимо уделять параметрам окружающей среды рабочей зоны. От температуры, давления и влажности зависит электробезопасность помещения. Кроме того, производственные метеоусловия в помещении существенно сказываются на качестве работы и производительности труда, а также на здоровье работающих.

Производственные метеоусловия рабочего места, оборудованного центральным отоплением и автоматической системой кондиционирования:

- Температура летом 22-24°С, зимой 20-22°С.
- Влажность 40%.
- Скорость движения воздуха 0,1 м/с.

Микроклимат рабочей зоны полностью соответствует нормам СанПиН 2.22.542-96, нормы представлены в таблице 14.

Таблица 14. Оптимальные и допустимые нормы микроклимата

.

Период года	Температура,°С	Относительная влажность, %	Скорость движения воздуха, м/с

	Оптимальная	Допустимая на рабочих местах				Оптимальная	Допустимая	Оптимальная, не более	Допустимая, не более
		Верхняя		Нижняя					
		Пост.	Не пост.	Пост.	Не пост.				
Холодный	22 - 24	25	26	21	18	40 - 60	75	0,1	0,1
Теплый	23 – 25	28	30	22	20	40 - 60	70	0,1	0,1

В помещении, где производилась работа, температура и влажность воздуха поддерживается в заданных в таблице пределах. Кроме того, имеется автоматическая система кондиционирования, очищающая и нагревающая (охлаждающая) поступающий в лабораторию воздух.

Таким образом, принятия дополнительных мер по созданию благоприятных условий не требуется.

5.1.3. Производственное освещение

Значение освещения в процессе жизнедеятельности, и особенно в производственной сфере - очень велико. При долговременной работе недостаточная освещенность рабочей зоны приводит к ослаблению зрительной активности и ухудшению зрения работающего. Правильно выполненная система освещения имеет большое значение в снижении производственного травматизма, уменьшая потенциальную опасность многих производственных факторов; создает нормальные условия для работы органам зрения и повышает общую работоспособность организма.

Согласно санитарно-гигиеническим требованиям рабочее место разработчика должно освещаться естественным и искусственным освещением.

Рабочая зона или рабочее место освещается в такой степени, чтобы можно было хорошо видеть процесс работы, не напрягая зрения, и чтобы исключалось прямое попадание лучей источника света в глаза. Кроме того,

уровень освещения определяется степенью точности зрительных работ. Наименьший размер объекта различения составляет 0.5 - 1 мм. Кроме того, в помещении отсутствует естественное освещение. По нормам освещенности СанНиП 23-05-95 (заменен СанНиП 23-05-2010) и отраслевым нормам, работа разработчика относится к четвертому разряду зрительной работы средней точности. Для этого разряда рекомендуется освещенность 400 лк.

Искусственное освещение осуществляется с использованием газоразрядных люминесцентных ламп низкого давления типа ЛБ-40, в количестве 9 светильников в каждом по 4 лампы. Произведем расчет искусственного освещения:

$$E = \frac{\Phi_{св} \cdot n \cdot N \cdot I}{K_з \cdot s \cdot z},$$

где E - номинальная освещенность рабочего места;

$\Phi_{св}$ - световой поток от лампы, лк;

N - количество светильников;

$K_з$- коэффициент запаса, учитывающий запыленность и износ светильников;

n - коэффициент использования светильников;

s - площадь помещения, $м^2$;

z - коэффициент неравномерности освещения.

Согласно СанНиП 23-05-2010 для использования данного типа ламп:

- $K_з$ = 1.4 при нормальной эксплуатации светильников;
- z = 1.2 при оптимальном размещении светильников.

$$I = \frac{A \cdot B}{h_C \cdot (A + B)},$$

где A - длина, м; A = 10 м;

B - ширина, м; B = 7 м;

h_c - высота светильников над рабочей поверхностью, h_c =3,2 м.

$$I = \frac{10 \cdot 7}{3,2 \cdot (10+7)} = 1,28.$$

По таблице 3 определяем коэффициент использования светильников n=0.4, при найденном значении I.

Таблица 16. Значения коэффициента использования светового потока в зависимости от показателя помещения

Показатель помещения, I	0.5	1	2	3	4
Коэффициент использования светового потока, h	0.22	0.36	0.48	0.54	0.59

Световой поток от лампы типа ЛБ-40 равняется 3120 лк. Тогда световой поток, излучаемый светильником, равняется 12480 лк.

Определим номинальную освещенность рабочего места:

$$E = \frac{12480 \cdot 90,4 \cdot 1,28}{1,4 \cdot 70 \cdot 1,2} = 489 лк.$$

Полученное значение соответствует условиям нормальной работы, следовательно, никаких дополнительных мероприятий проводить не нужно.

5.2 Анализ опасных производственных факторов

В качестве опасных факторов могут выступать механические травмы, термическое травмы, электрические травмы, возгорание. В процессе разработки программно-аппаратного комплекса системы технического зрения для автономного робота не возникнет никаких опасных факторов, связанных с механическим и термическим травмированием, так как отсутствуют какие-либо элементы, способные нанести вред человеку. Вследствие этого, рассмотрены электро-пожаробезопасность

5.2.1 Электробезопасность

Электрические установки, к которым относятся ЭВМ и измерительная аппаратура, представляют для человека большую потенциальную опасность. В процессе эксплуатации или при проведении профилактических работ человек может коснуться частей, находящихся под током.

3) Согласно классификации помещений по электробезопасности дипломный проект разрабатывался в помещении без повышенной опасности (класс 01 по ГОСТ 12.1.019–85), характеризующимся наличием следующих условий:

- напряжение питающей сети 220 В, 50 Гц;
- относительная влажность воздуха не более 75%;
- средняя температура не более 35°С;
- наличие непроводящего полового покрытия.

Основными техническими способами и средствами защиты от поражения током являются: защитное зануление, выравнивание потенциалов; защитное заземление, электрическое разделение сети, изоляция токоведущих частей, оградительные устройства и другое.

В помещении используются для питания приборов напряжение 220 В переменного тока с частотой 50 Гц. Это напряжение опасно для жизни, поэтому обязательны следующие предосторожности:

а) перед началом работы убедится, что выключатели, розетки закреплены и не имеют оголенных токоведущих частей;

б) не включать в сеть компьютеры и другую оргтехнику со снятыми крышками;

в) запрещается оставлять без присмотра включенное в электросеть оборудование;

г) при обнаружении неисправности компьютера необходимо выключить его и отключить от сети;

д) при обнаружении неисправностей или порчи оборудования необходимо, не делая никаких самостоятельных исправлений и ничего не разбирая сообщить преподавателю или ответственному за оборудование;

е) запрещается загромождать рабочее место лишними предметами;

ж) при несчастном случае необходимо немедленно отключить питание электроустановки, вызвать “СКОРУЮ ПОМОЩЬ” и оказать пострадавшему первую помощь до прибытия врача;

з) дальнейшее продолжение работы возможно только после устранения причины поражения электрическим током;

и) по окончании работы ответственный должен проверить оборудование, выключить все приборы.

5.2.2 Пожаровзрывобезопасность

По взрывопожарной опасности помещения и здания подразделяются на категории А, Б, В, Г, Д (ГОСТ12.1.004-91). Для используемого рабочего места установлена категория пожарной опасности - В (пожароопасное). Она характеризуется наличием горючих и трудно горючих жидкостей, твердых горючих и трудно горючих веществ и материалов, веществ и материалов, способных при взаимодействии с водой, кислородом или друг с другом только гореть при условии, что помещение, в котором они находятся, не относится к категориям А или Б.

В блоках аппаратуры, находящейся в помещении очень велика плотность размещения элементов электронных схем. В непосредственной близости располагаются соединительные провода, коммутационные кабели. Одной из наиболее важных задач пожарной профилактики, является защита строительных конструкций от разрушения и обеспечение достаточной прочности в условиях воздействия высоких температур при пожаре.

Здание, где находится используемое для работы помещение, построено из несгораемого материала - кирпича и относится к зданиям второй степени огнестойкости. Приведем возможные причины возникновения пожаров:

- наличие твердых горючих веществ;
- опасная перегрузка сетей, которая ведет за собой сильный разогрев токопроводящих проводников и загорания изоляции;
- различные короткие замыкания;
- пуск оборудования после ремонта.

Для предупреждения пожаров от коротких замыканий, перегрузок необходим правильный выбор монтаж и соблюдение установленного режима эксплуатации электрических сетей, дисплеев и других устройств.

Для предупреждения пожаров также необходимы следующие мероприятия:

- противопожарный инструктаж;
- соблюдение противопожарных норм и правил при установке оборудования, освещения;
- правильная эксплуатация оборудования;
- правильное размещение оборудования;
- современный профилактический осмотр, ремонт и испытание оборудования.

Для тушения пожаров можно применять: галоидированные углеводороды, углекислый газ, воздухо-механическую пену.

В здании на видном месте, вывешен план эвакуации при пожаре, а также пожарный щит с огнетушителями и с другим противопожарным оборудованием.

5.3. Региональная безопасность

Под региональной безопасностью понимают комплекс мер, предназначенных для ограничения отрицательного влияния человеческой деятельности на природу, или другими словами – охрана окружающей среды.

В ходе выполнения ВКР и дальнейшем использовании системы управления отсутствуют выбросы каких-либо вредных веществ в атмосферу, следовательно, не происходит загрязнения воздуха. Не происходит также каких-либо сбросов в водоемы, следовательно, не оказывается никакого влияния на гидросферу.

В свою очередь, во время выполнения дипломной работы образовывался различный твердый бытовой мусор, такой как использованные аккумуляторы, провода, канцелярские принадлежности и другие. Для уменьшения вредного влияния на литосферу необходимо производить сортировку отходов и обращаться в службы по утилизации для дальнейшей переработки или захоронения.

5.4. Организационные мероприятия обеспечения безопасности

Работа, выполняемая разработчиком, относится к умственной работе. По степени физической тяжести - к категории легких работ. Основная нагрузка падает на центральную нервную систему. При проектировании и организации оптимальных условий труда для разработчика должны быть соблюдены условия, позволяющие полноценно работать.

5.4.1. Технологические перерывы

По СанНиП 2.2.2.542-96 для обеспечения оптимальной работоспособности и сохранения здоровья разработка, на протяжении рабочей смены должны устанавливаться регламентированные перерывы. Время регламентированных перерывов в течение рабочей смены следует

устанавливать в зависимости от её продолжительности, вида работ и категории трудовой деятельности.

Мероприятия по снижения нервно-психологического напряжения и уменьшению его вредного влияния: установление рационального режима труда и отдыха, организация отдыха в процессе работы. Необходимо ввести нормированный 8 – часовой рабочий день: перерыв 20 минут каждые два часа.

5.4.2. Компоновка рабочего места

По СанНиП 2.2.2.542-96 схемы размещения рабочих мест с персональными компьютерами должны учитывать расстояния между рабочими столами и мониторами, которое должно быть не менее 2,0 метров, а расстояние между боковыми поверхностями мониторов не менее 1,2 метра.

Мероприятия по организации рабочих мест заключаются в следующем: необходимо вместо стандартных парт установить специальные столы с опорой для левой руки, с местом для размещения текстов и записей в зоне оптимальной досягаемости правой руки, с регулируемой по высоте клавиатурой и экраном монитора.

Заключение

Результатом проведенной работы стала спроектированная аппаратная и алгоритмическая составляющие системы управления для класса шагающих мобильных роботов. В данной работе эта система была рассмотрена применительно к роботу – гексаподу. Отличительной особенностью спроектированной системы является использование метода машинного обучения, называемого глубоким обучением с подкреплением, для генерации походки шагающего робота. Данный метод позволяет роботу осуществлять походку в стиле «от сенсоров к двигателям», без жесткого программирования алгоритма походки и без учета динамики и кинематики робота. Именно благодаря этому данный метод может быть использован для всего класса шагающих роботов.

В ходе работы также были исследованы методы локализации и построения карты местности. Был сделан вывод о том, что методы на основе фильтра частиц являются наиболее эффективными в задаче локализации и построении двухмерной карты типа «сетка занятости». Были выделены основные проблемы данного метода, а именно привязанность к колесному типу роботов, высокая ресурсоёмкость и высокая стоимость аппаратной реализации, и были предложены пути решения этих проблем.

Разработанные модули интегрируются вместе с модулем поиска кратчайшего пути в единую систему навигации с помощью операционной системы Robot Operating System. Программная реализация проводилась с помощью языков Python и C++. Использовались открытые библиотеки OpenCV, OpenNI, CUDA, TensorFlow, Caffe, ROS.

Список публикаций

1. Дусеев В. Р., Рудь М. Н., Мальчуков А. Н. Исследование методов фильтрации карты глубины с сенсора KINECT [Электронный ресурс] // Молодежь и современные информационные технологии: сборник трудов XI Международной научно-практической конференции студентов, аспирантов и молодых ученых, Томск, 13-16 Ноября 2013. – Томск: ТПУ, 2013 – С. 143–145.
2. Колчанова В. А., Рудь М. Н. Применение программно-интегрированных сред Mathcad, Matlab для исследования переходных процессов // Achievement of high school – 2012: материали за 8-а Международна научна практична конференция, София, 17-25 Ноября 2012. – София: «Бел Град-БГ» ООД, 2012. – Т. 26. Технологии – С. 16–18.
3. Курганов С. М., Рудь М. Н., Александрова Т. В. Разработка модели робототехнического комплекса по сортировке и транспортировке производственного мусора // Инженерные, научные и образовательные приложения на базе технологий National Instruments 2011: сборник по материалам X Международной научно-практической конференции, Москва, 8-9 Декабря 2011. – Москва: ДМК-пресс, 2011 – С. 270–271.
4. Рудь М. Н., Дусеев В. Р. Создание интерактивной системы визуализации [Электронный ресурс] // Современные техника и технологии: сборник трудов XIX Международной научно-практической конференции студентов, аспирантов и молодых ученых: в 3 т., Томск, 15-19 Апреля 2013. – Томск: ТПУ, 2013 – Т. 2 – С. 337-338.
5. Рудь М. Н. Контурный анализ в распознавании изображений [Электронный ресурс] // Молодежь и современные информационные технологии: сборник трудов X Международной научно-практической конференции студентов, аспирантов и молодых учёных, Томск, 13-16

Ноября 2012. – Томск: Изд-во Томского политехнического университета, 2012. – С. 255–256.

6. Rud M. N., Duseev V. R. Simulation of water flow over varying bottom using graphical processing units (GPUs) [Electronic resorces] // Ресурсоэффективным технологиям – энергию и энтузиазм молодых: сборник научных трудов V Всероссийской конференции студентов элитного технического образования, Томск, 25-27 Марта 2014. – Томск: Изд-во ТПУ, 2014 – С. 285–287.
7. Rud M. N. Use of Youbot System to Manipulate Objects [Electronic resorces] // Современные техника и технологии: сборник трудов XIX Международной научно-практической конференции студентов, аспирантов и молодых ученых: в 3 т., Томск, 15-19 Апреля 2013. - Томск: ТПУ, 2013 – Т. 2 – С. 447–448.
8. Дусеев В. Р., Рудь М. Н., Мальчуков А. Н. Исследование фильтра Калмана в приложении к фильтрации данных глубины с сенсора Kinect [Электронный ресурс] // Современные техника и технологии: сборник трудов XX Международной научно-практической конференции студентов, аспирантов и молодых учёных: в 3 т., Томск, 14-18 апреля 2014. - Томск: ТПУ, 2014 – Т.2 – С. 173–174.
9. Rud M. N., Duseev V. R. Simulation and visualization of water flow using graphical processing units (GPUs) [Electronic resorces] // Современные техника и технологии: сборник трудов XX Международной научно-практической конференции студентов, аспирантов и молодых учёных: в 3 т., Томск, 14-18 апреля 2014. – Томск: ТПУ, 2014 – Т.2 – С. 303–304.
10. Рудь М. Н. Моделирование переходных процессов в электрических цепях [Электронный ресурс] // Энергетика: эффективность, надёжность, безопасность: труды XIV международного студенческого научно-

технического семинара: в 2 т., Томск, 24-27 апреля 2012. – Томск: ТПУ, 2012 – Т.2 – С. 337–339.

11. Рудь М.Н. Искусственные мышцы: технология и перспективы [Электронный ресурс] // Современные методы механики: сборник трудов международной молодёжной конференции: Томск, ТГУ, 19-20 сентября 2012. – Томск, ТГУ, издательство Томского университета, 2012 – С. 77–79.
12. Rud M. N. , Pantyukhin A. R. Development of GPU-accelerated localization system for autonomous mobile robot // Mechanical Engineering, Automation and Control Systems: Proceedings of International Conference, Tomsk, October 16-18, 2014. - Tomsk: TPU Publishing House, 2014 - p. 1-4
13. Рудь М. Н. Система локализации для автономного мобильного робота с использованием технологии CUDA // Современные информационные технологии и ИТ-образование. - 2014 - №. 1(9). - С. 555-563
14. Рудь М. Н. , Семенов Н. М. Разработка системы обучения с подкреплением для шагающего робота // Перспективные системы и задачи управления: материалы Одиннадцатой Всероссийской научно-практической конференции и Седьмой молодежной школы-семинара "Управление и обработка информации в технических системах", Ростов-на-Дону, 4-8 Апреля 2016. - Ростов-на-Дону: ЮФУ, 2016 - Т. 1 - С. 247-259
15. Рудь М. Н. , Пантюхин А. Р. Использование графических процессоров при разработке интеллектуальных роботов [Электронный ресурс] // Искусственный интеллект: философия, методология, инновации: сборник трудов VIII Всероссийской конференции студентов, аспирантов и молодых ученых. Часть 1. Секции 1-5, Москва, 20-22 Ноября 2014. - Москва: МГТУ МИРЭА, 2014 - С. 161-166.

Список используемых источников

1. M. Anthony Lewis, Andrew H. Fagg and George A. Bekey. Genetic algorithms for gate synthesis in hexapod a robot. *Recent Trends in Mobile Robots, pp 317-331, World Scientific, New Jersey, 1994*
2. Gary B. Parker, David W. Braun, Ingo Cyliax. Evolving hexapod gaits using cyclic genetic algorithm. *Indiana University, 2000*
3. I. Kecskes, E. Burkus, F. Bazso. Optimization of the hexapod robot walking by genetic algorithm. *Intelligent Systems and Informatics (SISY), 2010 8th International Symposium, pp 121-126, IEEE, 2010.*
4. Mohammadali Shahriari. Design, implementation and control of hexapod robot using reinforcement learning approach. *Kish Island, Iran, 2013.*
5. Hrdlicka Ivo, Kutilek Patrik. Reinforcement learning in control systems for walking hexapod robots. *University of Brno, 2005.*
6. Nick Vallidis. A Hexapod Robot and Novel Training Approach for Artificial Neural Networks.
7. J. Zico Kolter, Youngjun Kim, Andrew Ng. Stereo Vision and Terrain Modeling for Quadruped Robots. *Proceedings of the IEEE International Conference on Robotics and Automation*, 2009.
8. Mnih, Volodymyr, Kavukcuoglu, Koray, Silver, David, Rusu, Andrei A, Veness, Joel, Bellemare, Marc G, Graves, Alex, Riedmiller, Martin, Fidjeland, Andreas K, Ostrovski, Georg, et al. Human-level control through deep reinforcement learning. *Nature, 518(7540):529–533, 2015.*
9. Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. *ArXiv e-prints, September 2015.*
10. Levine Sergey, Finn Chelsea, Darrell Trevor, and Abbeel Pieter. End-to-end training of deep visuomotor policies. *arXiv preprint arXiv:1504.00702, 2015.*

11. Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. Imagenet classification with deep convolutional neural networks. *In Advances in neural information processing systems, pp. 1097–1105, 2012.*
12. Daniel Maturana, Sebastian Scherer. VoxNet: A 3D Convolutional Neural Network for Real-Time Object Recognition. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, September 2015.
13. Jonathan Masci, Ueli Meier, Dan Ciresan, and Jurgen Schmidhuber. Stacked Convolutional Auto-Encoders for Hierarchical Feature Extraction. *Artificial Neural Networks and Machine Learning – ICANN 2011, pp 52-59.*
14. Hafner, Roland and Riedmiller, Martin. Reinforcement learning in feedback control.
Machine Learning, 84(1-2):137–169, 2011. ISSN 0885-6125. doi: 10.1007/s10994-011-5235-x.
15. Hausknecht, Matthew J. and Stone, Peter. Deep recurrent q-learning for partially observable mdps. *CoRR, abs/1507.06527, 2015*
16. Kingma, Diederik P., Ba, Jimmy. Adam: A method for stochastic optimization. *CoRR, abs/1412.6980, 2014*
17. Masson, Warwick and Konidaris, George. Reinforcement learning with parameterized actions. *CoRR, abs/1509.01644, 2015*
18. Oh, Junhyuk, Guo, Xiaoxiao, Lee, Honglak, Lewis, Richard L., and Singh, Satinder P. Action-conditional video prediction using deep networks in atari games. *CoRR, abs/1507.08750, 2015.*
19. Riedmiller, Martin A. and Gabel, Thomas. On experiences in a complex and competitive gaming domain: Reinforcement learning meets robocup. *In CIG, pp. 17–23. IEEE, 2007. ISBN 1-4244-0709-5.*
20. Stadie, Bradly C., Levine, Sergey, and Abbeel, Pieter. Incentivizing exploration in reinforcement learning with deep predictive models. *CoRR, abs/1507.00814, 2015*

21. Морзеев Ю.. Технологии машинного зрения. Сделано в России. // Компьютер – пресс. 2012. № 7. URL: http://compress.ru/article.aspx?id=11273.
22. NVIDIA VisionWorks: [Электронный ресурс] / NVIDIA. URL: http://www.nvidia.com/object/visionworks-cv-toolkit.html, свободный. – Загл. с экрана. – Яз. англ. Дата обращения: 10.06.2014 г.
23. Alluse Y. GpuCV: A GPU – accelerated framework for image processing and computer vision [Электронный ресурс] // URL: http://www-public.int-edu.eu/~horain/Publications/ISCV08/ISCV08_Allusse.pdf
24. Ramos E. Arduino and Kinect Projects // Apress, Technology in Action, 2011, 314 p.
25. Встраиваемая платформа разработок Jetson TK1 [Электронный ресурс] / NVIDIA. URL: http://www.nvidia.ru/object/jetson-tk1-embedded-dev-kit-ru.html, свободный - Загл. с экрана. – Яз. рус. Дата обращения: 29.04.2014 г.
26. Telea A. An Image Inpainting Technique Based on the Fast Marching Method [Электронный ресурс] // Eindhoven Institute of Technology. URL: http://iwi.eldoc.ub.rug.nl/FILES/root/2004/JGraphToolsTelea/2004JGraphToolsTelea.pdf
27. Brahmbhatt S. Practical OpenCV // Apress, Technology in action, 2012, 231 p.
28. Parker J. R. Algorithms for image processing and computer vision 2nd edition // Willey Publishing, 2011. 454 p.
29. Kinect Fusion [Электронный ресурс] / Microsoft. URL: http://msdn.microsoft.com/en-us/library/dn188670.aspx, свободный, - Загл. с экрана. – Яз. англ. Дата обращения: 25.03.2014 г.
30. Using kinfu large – scale to generate a texture mesh [Электронный ресурс] / PCL. URL:

http://pointclouds.org/documentation/tutorials/using_kinfu_large_scale.php, свободный - Загл. с экрана. – Яз. англ. Дата обращения: 25.03.2014 г.

31. Сандерс Дж. Технология CUDA в примерах, введение в программирование графических процессоров // АМК, Москва, 2013, 246 с.
32. Programming guide :: CUDA Toolkit documentation [Электронный ресурс] // URL: http://docs.nvidia.com/cuda/cuda-c-programming-guide/#axzz34kxz8Yel.
33. Shreiner D. OpenGL Programming guide 8th edition // Addison – Wesley, 2013, 346 p.

Walking gate generation with deep reinforcement learning

Reinforcement Learning (RL) techniques are interesting subjects in both control theory and cognitive sciences. In control theory, building a system that works perfectly is quite difficult, and it is an exhaustive procedure when unexpected errors or disturbances affect the system. By building a system that learns how to accomplish a task on its own, there is no need to calculate and predict complex control algorithms. In cognitive sciences, the ability to learn is a core component of cognition.

Reinforcement learning algorithm is one such simple learning algorithm. This section explores the ability of a robotic hexapod agent to learn how to walk, using only the ability to move its legs and tell whether the robot is moving forward. Therefore, the hexapod may be seen as an analog for a biological subject interacting with environment and having no external support or parental figure to learn from.

In supervised learning, the targets, i.e. right answers, are given to the agent as the training set. In some control problems, it is difficult or not feasible to define the exact correct supervision. For example, in hexapod walking, it is hard to define explicit supervision that the algorithm of learning is trying to mimic. In reinforcement learning instead of telling the exact supervision, only a reward function shows that the agent is doing well or not. Therefore, it will be a job of learning algorithm to find the best actions, which lead to larger rewards. There will be no need to define input/output sets. This kind of learning focuses on online performance and the interaction between exploration and exploitation. The trade between these two has been one of the most challenges, which have been studied in recent years [29]. It has been shown that reinforcement learning has different successful applications in autonomous systems and legged robot locomotion. Q-learning is a simple approach of Reinforcement Learning which has been chosen for walking learning. To define the problem, here, firstly, actions and states of problem

should be defined. The robot explores state action domain and gets reward in every possible action. In every step, from every state of the agent, each action is rewarded and the rewards of specific state action is stored. Future actions are taken based on the actions done and specified rewards in a way, which maximizes the coming reward in the present and future states and actions. It is desired here to learn the time sequencing of hexapod gait. The legs are moved in possible states with different specified actions. At the end of learning procedure, the sequence of actions in different states should be learned satisfying the goal, which is walking on a straight line while minimizing tilt and other undesired translations.

To teach robot to walk in different conditions, we have to define parameters, which will characterize both state of the environment (terrain) and internal state of robot. In this case, RGBD sensor Microsoft Kinect is the main source of visual information about the environment. This sensor provides information both about terrain appearance (RGB-frame) and its geometric structure (D-frame). As for internal state of robot, it consists of states of all legs. Each leg can be in three different states. First state – leg is above the ground. Two other states – leg is in front or behind of initial position (Figure 1):

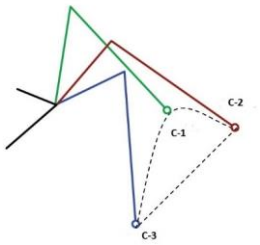

Figure 1 – Possible states of robot leg

Structure of model for processing RGB frame is shown on Figure 2:

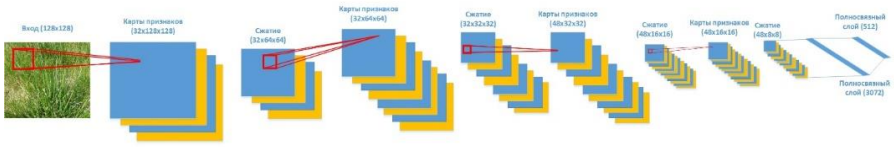

Figure 2 - Structure of convolutional neural network for RGB frame processing

To learn high – level visual features of terrain from RGB frame we use *convolutional neural network*. Main part of such types of models is convolutional layer that applies convolution operation to its input and generates so-called *feature maps* as output. After convolutional layer there is a nonlinearity, that is applied to feature maps. In convolutional neural networks nonlinearity of type «rectified linear unit» is widely used (values below zero are equated to zero). Such type of nonlinearity is used in this work. Optional element of convolutional neural netwok is *pooling layer*. This layer decreases size of image by averaging or maximizing pixel intensity in some defined neighborhood (pooling area). Application of pooling can help to achieve invariance in translation. Usually in convolutional neural networks from two to several dozens of combinations «convolution – nonlinearity – pooling» are used. These combinations follow one after another. After them there are one or several fully-connected layers, that give an ability to aggregate learned features. Outputs of the last fully-connected layer can be used to solve classification or regression problem or to go for further processing. In our case these outputs become a part of description of state «robot and environment».

Convolutional neural network that is used in this work consists of three combinations «convolution – nonlinearity – pooling», which are followed by two fully – connected layers. We use convolutional layer of type «same». It means that after convolution size of image does not change.

Pooling is applied with pooling size 2x2 by choosing pixel of maximum intensity. Therefore, after each stage size of image decreases by factor of 2 for both dimensions.

We apply model with the same structure for D-channel. We uses special pre-processing of depth frame, transforming it into RGB-image. Depth value now corresponds to combination of R,G,B values, where red color corresponds to the most distant pixel, and blue color to the nearest one. It is important to mention, that before processing depth frame is filtered with gradient filter [19] for in-painting of areas that are not available for measurements and with median filter to eliminate small noisy areas.

Structure of convolutional neural network for depth channel processing is shown on Figure 3:

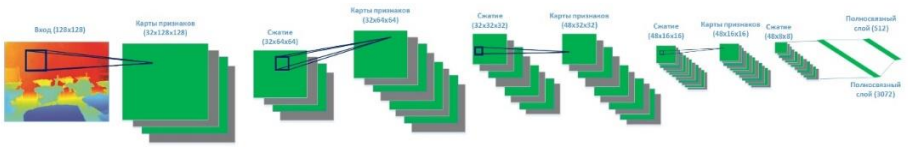

Figure 3 - Structure of convolutional neural network for depth frame processing

After data flows along the depth of both networks, we need to aggregate learned features (outputs of the last fully-connected layer) and to add values of internal robot state. To do this, we bind these three vectors vertically.

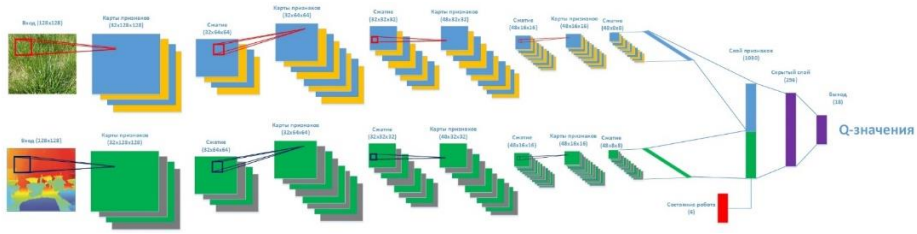

Figure 4 – Final network structure

As we can see from figure 4, input of each part of model is corresponding part of data about robot and environment state. Each part of these data flows through its own transformations along the network, and on aggregation stage obtained vectors are connected with each other. After that there is one more fully-connected layer, which is followed by output layer. Outputs are so-called Q-values, i.e. values of usefulness of each possible action (transitions between states). This part will be described in detail in following sections.

Algorithm for deep reinforcement learning

Reinforcement learning differs from supervised learning, because during the learning process agent does not receive correct answers, but rewards, that allow agent to understand, how well it fulfills determined task. Therefore, the goal of learning algorithm is to find behavior strategy that maximizes reward. One more feature of reinforcement learning is that learning itself happens on-line, during interaction of agent with environment.

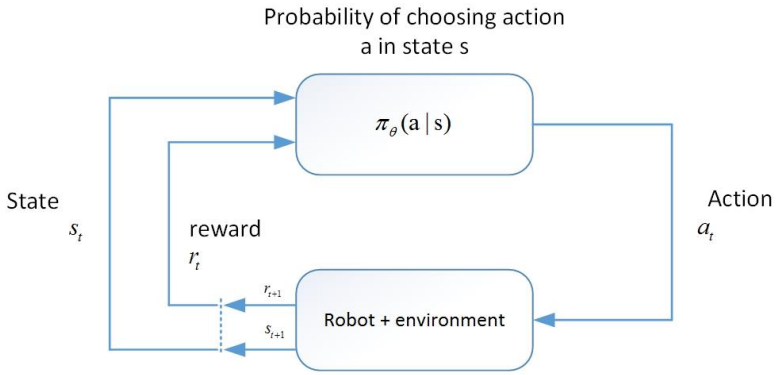

Figure 5 – Reinforcement learning structure

$$\max_{\theta} \ \mathrm{E}[\sum_{t=0}^{H} R(s_t)|\pi_\theta]$$

To describe learning process, we need firstly to define state and action domain. Robot searches across state-action domain and receives reward for each action. Future actions are fulfilled based on previous ones and on rewards received.

As was described in previous section, state of robot is set of leg states. Each leg can be in one of three states, so robot state can be described by 6 parameters. Action set consists of all possible transitions between states for each leg (36 actions, 6 for each leg). Of course, probability of choosing some of these actions is small. For example, when the leg is in state 1 (above the ground), transition back to state 3 seems to be not very useful, or, if leg is in state 2, transition back to 1 will not help robot to walk forward, but, nevertheless, we include these actions in set of available ones. It is possible, that in process of learning to walk on rough terrain, some of these actions will turn to be useful.

To determine angles of servo rotations, that are needed to perform transitions between states, we need to solve inverse kinematics problem. In this case, when state domain is discrete and finite, we suggest to obtain angles firstly, for each possible state. Using this method, we will create a table with angles corresponding to each transition. This table will be used to form control signals to servos.

One of the most important problems of reinforcement learning is defining reward function. Following assumption, that criteria of performing task well is forward distance with simultaneous minimization of tilt, the most obvious and effective solution will be using following formula:

$$R(s,a) = \frac{dist\_forward}{\sqrt{translation^2 + tilt^2}}$$

Tilt of robot is measured with inertial measurement unit (IMU). Changes in linear coordinates are measured with help of visual odometry system, which also uses Kinect sensor measurements.

Now we will describe algorithm of neural network learning. In this work we use special type of reinforcement learning, that is called Q-learning. In Q-learning we define function *Q(s,a)*, that represents maximum future reward in condition of choosing action *a* in state *s*:

$$Q(s_t, a_t) = \max R_{t+1}$$

The most convenient way to think about *Q(s,a)* – it is the most profitable score in the end of the game after fulfilling action *a* in state *s*.

«Game» in this case can represent vast variety of different tasks, that robot can be forced to accomplish. This function is called Q-function, because it represents Quality of certain action in certain state.

Knowing Q-value for each action we can determine strategy of agent behavior :

$$\pi(\mathrm{s}) = \mathrm{argmax}_a\, Q(s, a)$$

This formula corresponds to so-called «greedy» algorithms of action selection. However, there are some problems connected with this algorithms. It is important to mention here, that one of the main problems of reinforcement learning is to find balance between exploration and exploitation. This problem means, that when agent finds suitable behavior strategy, it can stop search, assuming, that strategy found is the best one and not fulfilling actions, that are far from this strategy. This can lead to the situation, that agent will lose the chance to find the best possible strategy.

To avoid such a problem, we can use so-called ε- greedy learning algorithm. When using this algorithm, with probability ε arbitrary action can be selected, and action with maximum Q-value is selected with probability 1 - ε. When implementing ε-greedy algorithm following situation can occur: at the end of learning process, when

agent found the best strategy and Q-values converged to optimal ones, agent selects arbitrary action or sequence of arbitrary actions that can lead to destabilization and divergence of learning. To avoid such problems, value of ε is not fixed during learning process, but it decreases over time. In this case, when agent is close to the end of learning process, probability of choosing arbitrary action is low.

Now we will show how to obtain Q-values. We consider only one transition $< s, a, r, s' >$, where s – current state, a – selected action, r – obtained reward, s' – next state after performing action a.

$$Q(s,a) = r + \gamma \max_{a'} Q(s', a')$$

This equation is so-called Bellman equation. It is quite logical – maximum future reward for certain state equals to the immediate reward plus maximum future reward for the next state. Parameter γ represents, how much attention is for future rewards in comparison with immediate one. The main idea of Q-learning is the ability to iteratively approximate Q-function using Bellman equation. In the simplest case Q-function is represented by a table with states in rows and actions in columns. Algorithm of Q-learning is presented below:

Algorithm 1. Classical Q-learning

Initialization Q[N_states, N_actions] with arbitary values

Obtain data about initial state ***s***

Repeat

Choose and perform action ***a***

Receive information about reward **r** and about new state ***s'***

$Q(s,a) = Q(s,a) + \alpha(r + \gamma \max_{a'} Q(s', a') - Q(s,a))$

s = ***s'***

until termination of episode

«Episode» in this case represents a sequence of transitions between initial state and terminal state. Terminal state can occur when robot faced with an obstacle, lost stability or made no progress in movement during certain period of time. α in this algorithm – learning rate, that controls, how fast learning process will be performed. If $\alpha = 1$, then two components *Q(s,a)* compensate each other and we obtain Bellman equation.

If we represent Q-function as a table in this task, we face so-called problem of dimensionality. If we count only internal state of robot, which can be represented as a set of leg states, we can describe it with discrete variables and can create Q-table. But having visual data from sensor as input, we have infinite number of possible states, and, taking this fact into account, we need to use different representation ща Q-function. In this case, we use deep neural network.

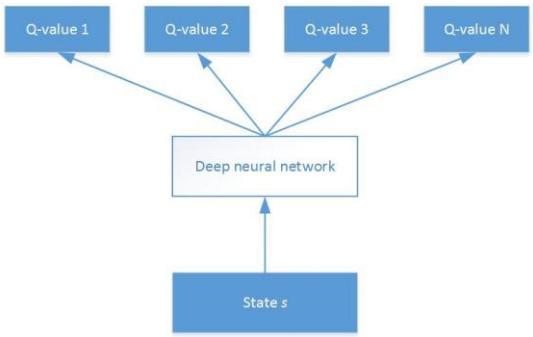

Figure 6 – Approximation of Q-function with deep neural network

Q-function can have any value, what means that we will teach neural network to perform a regression task.

Optimization problem can be solved with the following loss function:

$$L = \frac{1}{2}[r + \gamma \max_{a'} Q(s', a') - Q(s, a)]^2$$

Assuming transition $< s, a, r, s' >$, learning algorithm will have the following structure:

Algorithm 2. Deep Q-learning algorithm

8. Forward pass through the network for current state *s* to obtain predicted Q-values for actions.
9. Selection of action with maximum Q-value or in accordance with ε - greedy algorithm.
10. Receiving information about reward *r*
11. Simulation of transition from *s* to the next state *s'*.
12. Forward pass through the network for the next state *s'* and computation of $\max_{a'} Q(s', a')$.
13. Computation of $r + \gamma \max_{a'} Q(s', a')$ for action from step 2. For all other actions set Q-value equal to original one, obtained on step 1, assuming error for corresponding outputs equal to zero.
14. Updating of neural network weights using back-propagation.

Deep Q-learning in continuous action domain

It is important to mention, that deep Q-learning algorithm, that was described in previous sections, works in continuous space domain, but in discrete action domain. Each leg of hexapod robot can be in one of three discrete states, so robot can be in one of $3^6 = 729$ states. Action space is formed from all possible transitions between states. This approach has its advantages and disadvantages. The main

advantage is simplicity of Q-learning implementation. The main disadvantage is the lack of opportunity to explore such actions as moving robot body up and down, increase of step width, side walk, i.e. all actions that require being in states, different from which were determined.

More effective variant is the model, that assumes transition from continuous input data about state directly to servo rotation angles. In such model robot, having no knowledge about possible actions and without using inverse kinematics, would have an ability to find an optimal walking strategy. Unfortunately, Q-learning in its classical variant, including deep Q-learning, is not suitable for solving continuous control task, because the aim of Q-learning is to provide Q-values for finite number of actions.

To perform reinforcement learning in continuous action domain (continuous control), we suggest to use architecture Actor – Critic. This architecture assumes diversification of Q-value computation and action selection. Represented by one deep network, actor provides continuous action as output, and critic, that is represented by second deep network, evaluates Q-value for this action.

Actor netword μ, parameterized by θ^{μ}, takes state *s* as input and outputs continuous action *a* (in our case, set of servo angles). Critic network *Q*, parameterized by θ^{Q}, takes state *s* and action *a* as input, and outputs scalar *Q*-value *Q(s,a)*.

Method of updating critic network weights stays the same as was described in section 3:

$$Q(s,a) = Q(s,a) + \alpha(r + \gamma \max_{a'} Q(s',a') - Q(s,a))$$

For neural network case:

$$L_Q(s,a \mid \theta^Q) = (Q(s,a \mid \theta^Q) - (r + \gamma \max Q(s',a' \mid \theta^Q)))^2$$

However, this equation is not suitable for continuous action domain, because it includes maximization of Q-values for the next state. Instead of this, we ask actor to

provide us next action $a' = \mu(s', \theta^{\mu})$. It leads to the following loss function for the critic:

$$L_Q(s, a \mid \theta^Q) = (Q(s, a \mid \theta^Q) - (r + \gamma Q(s', \mu(s' \mid \theta^{\mu}) \mid \theta^Q)))^2$$

Approximation of critic Q-function can be obtained by optimization using gradient descent in relation to parameters θ^{μ}. However, adequacy of this model depends on quality of strategy chosen by actor, because actor determines next action *a'*, that we use to update critic network weights.

Critic knowledge about Q-values of actions can be used to teach actor strategy. Having certain state, actors goal is to minimize difference between current selected action *a* and optimal action *a** in this state:

$$L_{\mu}(s \mid \theta^{\mu}) = (a - a^*)^2 = (\mu(s \mid \theta^Q) - a^*)^2$$

Critic can be used to provide information about quality of different actions, but direct evaluation of *a** will lead to maximization of critics output along all possible actions:

$$a^* \approx \operatorname{argmax}_a Q(s, a \mid \theta^Q)$$

Instead of using such global maximum, critic network can provide gradients, that can show the direction of changes (in action domain), that will lead to greater Q-values: $\nabla_a Q(s, a \mid \theta^Q)$. We can update these gradients with one backward pass through the critic network, that is much faster, then solving optimization task in continuous action domain. It is important to mention, that these gradients are obtained not in relation to parameters θ^Q, but in relations to inputs. To update actor weights, we need to use these gradients as actor output layer values and perform one backward pass through actor network.