|  |
| --- |
| План-проспект дипломной работы  Тема: **РЕАЛИЗАЦИЯ И ИССЛЕДОВАНИЕ АЛГОРИТМА ГЕНЕРАЦИИ ТРАЕКТОРИЙ В ДИНАМИЧЕСКОЙ СРЕДЕ НА ОСНОВЕ OCTNet**  Финальный объем – 80 стр. |

**Содержание**

[1. Введение (6 стр.) 5](#_Toc69394860)

[1.1. Актуальность 5](#_Toc69394861)

[1.2. Проблема 5](#_Toc69394862)

[1.3. Объект исследования 6](#_Toc69394863)

[1.4. Цель 6](#_Toc69394864)

[1.5. Задачи 6](#_Toc69394865)

[2. Обзор аналогов (10 стр) 6](#_Toc69394866)

[2.1. Аналоги системы 6](#_Toc69394867)

[2.2. Модели работающие с фото 7](#_Toc69394868)

[2.3. Модели работающие с видео 7](#_Toc69394869)

[2.4. Модели работающие с картами 7](#_Toc69394870)

[2.5. Критерии сравнения 8](#_Toc69394871)

[2.6. Сравнение аналогов 8](#_Toc69394872)

[3. OCTNet (5 стр) 9](#_Toc69394873)

[3.1. Архитектура системы 9](#_Toc69394874)

[3.2. Входные данные 9](#_Toc69394875)

[3.3. Модуль предобработки 10](#_Toc69394876)

[3.4. Модуль обучения модели 10](#_Toc69394877)

[3.5. Модуль генератора траекторий 11](#_Toc69394878)

[4. Статическая модель OCTNet (15 стр) 12](#_Toc69394879)

[4.1. Данные 12](#_Toc69394880)

[4.2. Представление траекторий 12](#_Toc69394881)

[4.2.1. Дискретные 12](#_Toc69394882)

[4.2.2. Непрерывные 12](#_Toc69394883)

[4.3. Признаки модели 12](#_Toc69394884)

[4.4. Архитектура модели 12](#_Toc69394885)

[4.5. Предсказания модели 12](#_Toc69394886)

[4.6. Генерация траектории по предсказанию 12](#_Toc69394887)

[5. Динамическая модель OCTNet (20 стр) 13](#_Toc69394888)

[5.1. Данные 13](#_Toc69394889)

[5.2. Представление траекторий 13](#_Toc69394890)

[5.2.1. Дискретные 13](#_Toc69394891)

[5.2.2. Непрерывные 13](#_Toc69394892)

[5.3. Признаки модели 13](#_Toc69394893)

[5.4. Архитектура модели 13](#_Toc69394894)

[5.5. Предсказания модели 13](#_Toc69394895)

[5.6. Генерация траектории по предсказанию 13](#_Toc69394896)

[6. Сравнения точности работы алгоритмов (15 стр) 15](#_Toc69394897)

[6.1. Сравнение статических моделей 15](#_Toc69394898)

[6.1.1. GAN 15](#_Toc69394899)

[6.1.2. CVAE 15](#_Toc69394900)

[6.1.3. KTM 15](#_Toc69394901)

[6.1.4. OCTNet 15](#_Toc69394902)

[6.2. Сравнение динамических моделей 15](#_Toc69394903)

[6.2.1. GAN 15](#_Toc69394904)

[6.2.2. CVAE 15](#_Toc69394905)

[6.2.3. KTM 15](#_Toc69394906)

[6.2.4. OCTNet 15](#_Toc69394907)

[7. Оценка и защита результатов интеллектуальной деятельности (10 стр) 16](#_Toc69394908)

[Заключение (1 стр) 16](#_Toc69394909)

[Литература 17](#_Toc69394910)

[Приложение 18](#_Toc69394911)

# Введение (6 стр.)

Беспилотные автомобили, беспилотная строительная техника и роботы- помощники сегодня активно развиваются и находят свое применение как в жизни обычных людей, так и на разнородных промышленных и военных предприятиях.

## Актуальность

Для успешной и качественной работы беспилотных машин им необходимо принимать решения со скоростью, превышающей скорость работы человеческого мозга. Исходя из этого, алгоритмы, использующиеся для управления этими устройствами, должны работать очень быстро. Одним из критических алгоритмов для подобных устройств является алгоритм генерации траекторий. Для устройств, движение которых будет происходить среди обычных людей, правильная траектория движения критически важна. Выбор управляющим устройством аварийной траектории может привести к аварийным последствиям.

## Проблема

Для решения проблемы генерации траектории на текущий момент используются два семейства алгоритмов: алгоритмы на графах и алгоритмы искусственного интеллекта. Основным алгоритмом на графе для решения проблемы выступает алгоритм A\* (A star). Из наиболее популярных алгоритмов искусственного интеллекта можно привести в пример: GAN и CVAE [3]. Алгоритмы на графах не могут учитывать быстрое изменение окружающей обстановки и не имеют способности учитывать предыдущий опыт. В свою очередь приведенные в пример алгоритмы машинного обучения работают достаточно точно, однако не имеют возможности получать информацию из динамической обстановки среды. Из-за отсутствия знаний о характере динамики внутри среды, существующие решения в сфере искусственного интеллекта будут работать менее быстро или будут тратить больше ресурсов, по сравнению с разрабатываемым алгоритмом. Помимо прочего, разрабатываемое решения имеет более высокую точность, чем аналоги. Наличие данной проблемы диктует описанную далее цель.

## Объект исследования

## Цель

Цель – реализовать программный модуль, который позволит генерировать возможные траектории маршрутов быстрее, чем существующие аналоги и будет эффективно работать в динамических системах. Сформулируем задачи для достижения цели.

## Задачи

Для достижения заданной цели требуется решить следующие задачи:

* Провести анализ аналогов;
* Разработать алгоритм подготовки входных данных для сети;
* Разработать алгоритм сети для статической системы OCTNet [4];
* Разработать алгоритм сети для динамической системы;
* Провести сравнение замеров точности с аналогами.

# Обзор аналогов (10 стр)

## Аналоги системы

Среди аналогов системы необходимо рассмотреть как метод поиска пути в графе, так и методы, использующие машинное обучение. Основными конкурентами алгоритма являются A\*, KTM, GAN, CVAE. Опишем кратко каждый из них.

## Модели работающие с фото

## Модели работающие с видео

## Модели работающие с картами

### A\*

А\* - алгоритм поиска кратчайшего пути в графе. Если представить все пространство как граф, каждый узел которого соединен со всеми своими смежными соседями, то можно использовать его для поиска маршрута в этом графе. Плюсом такого подхода является простота реализации, а минусами является время работы по сравнению с алгоритмами машинного обучения, необходимость модифицировать правила для каждой новой среды и низкое качество работы в динамических средах.

### KTM

KTM - модель машинного обучения, реализованный в виде непрерывной функции. Умеет строить маршрут в конкретной среде, после того как обучится работать в подобных средах. Минусом является неспособность работать в неизвестных средах.

### GAN

GAN - генеративная модель, которая может быть приспособлена к построению траекторий. Успешно показала себя в семействе генеративных задач. Минусом данной модели является необходимость приспосабливать ее для данной задачи и, как следствие, неспособность работать в неизвестных и динамических средах.

### CVAE

CVAE также является генеративной моделью, со всеми минусами и плюсами GAN. Было решено рассматривать данную модель как конкурента, поскольку GAN и CVAE обучены по-разному и могут дать различные результаты в поиске траекторий.

### OCTNet

OCTNet - генеративная модель, учитывающая данные об окружающей среде. Данный алгоритм спообен строить корректные маршруты как в известных, так и в неизвестных средах. Помимо прочего, алгоритм умеет строить траекторию в каждый момент маршрута, что делает его способным строить маршрут в динамической среде. Алгоритм требует лишь ресурсы для хранения модели, по которой происходит потроение.

Выберем критерии сравнения аналогов.

## Критерии сравнения

Для алгоритма поиска траекторий важны следующие свойства:

* Умение работать в известных средах;
* Умение работать в неизвестных средах;
* Умение работать в динамических средах;
* Способность быстро генерировать траекторию в каждый момент времени;
* Умение работать в условиях ограниченных ресурсов.

Основываясь на данных критериях, перейдем к сравнению аналогов.

## Сравнение аналогов

В таблице Табл. 1 представлено сравнение аналогов по указанным ранее критериям.

Таблица 1 - сравнение аналогов

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Название** | ***Известные средаы*** | ***Неизвестные средаы*** | ***Динами-ческие среды*** | ***Непре-рывность*** | ***Ограни-ченные ресурсы*** |
| A\* | + | - | - | - | - |
| KTM | + | - | + | + | - |
| GAN | + | - | - | + | + |
| CVAE | + | - | - | + | + |
| OCTNet | + | + | + | + | + |

Из данных в Табл. 1 видно, что существующие решения не решают описанную ранее проблему или решают ее не в полной мере. Исходя из этого, можно подтвердить актуальность данной работы

# OCTNet (5 стр)

## Архитектура системы

Архитектура программной реализации состоит из 3 модулей:

* Модуль предобработки данных
* Модуль обучения модели
* Модуль генератора траекторий

## Входные данные

Прежде чем говорить об архитектуре системы, необходимо определиться с входными данными. Для обучения модели будет использоваться набор данных - Occ-Traj 120. Данный набор состоит из нескольких тысяч карт расстановок с размеченными маршрутами. Часть набора для обучения содержит расстановки для траекторий, а часть для тестирования расстановок для траекторий не содержит. Данные о карте в наборе Occ-Traj 120 имеют следующий вид:

[[1 1 1 1 0 0 0 1 1 1] [1 1 1 1 0 0 0 1 1 1] ......

[1 1 1 1 1 1 1 1 1 1] [1 1 1 1 1 1 1 1 1 1]]

Где:

означает место, по которому можно пройти

1 - место по которому нельзя пройти.

Данные траекторий имеют структуру, в которой сначала идет id траектории, за которым следует список пар координат.

## Модуль предобработки

Программа загружает из сети dataset с набором данных для обучения и валидации модели. Данные в данном наборе не удобны для работы, поэтому необходимо провести конвертацию и валидацию данных. Поскольку набор данных представляет из себя файл с данными размером в несколько гигабайт, необходимо реализовать обработчик, который будет поддерживать кеширование и запоминать место остановки в случае ошибки. На данный момент, реализованный ранее парсер не поддерживает восстановление после отказа, поэтому его необходимо будет доработать. Предполагаемая модель модифицированного парсера - потоковый парсер с указателем на последний считанный элемент набора данных. Архитектура парсера с доработками представлена на рис.1:

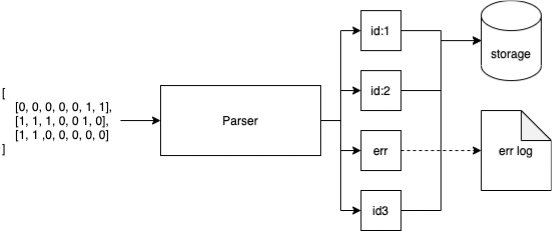


Рисунок 1 - потоковый парсер с обработкой отказов

## Модуль обучения модели

На вход модулю подается карта расстановки преград и коридоров, которая получается на выходе работы парсера, а также допустимые траектории. Модуль переводит данные в вектора признаков, которые подаются на вход алгоритму MDN для обучения модели. Архитектура модуля обучения изображена на рис. 2:

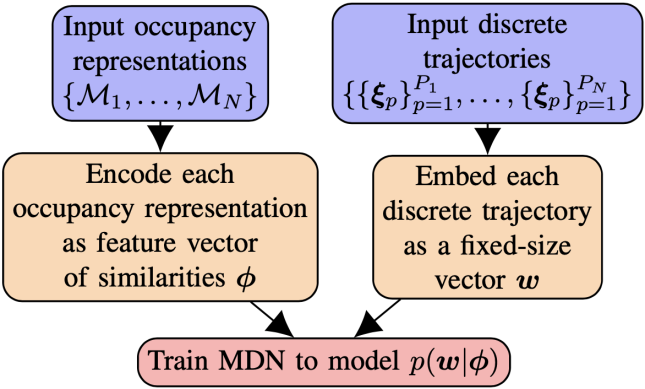


Рисунок 2 - архитектура модуля обучения модели

## Модуль генератора траекторий

На вход OCTNet подается вектор признаков, описывающий расстановку, по которому обученная модель умеет выдавать предполагаемую траекторию. Траектория может быть не корректной, поэтому модуль предполагает валидацию траектории с возможностью повторного перерасчёта. Архитектура модуля генератора траекторий изображена на рис. 3:

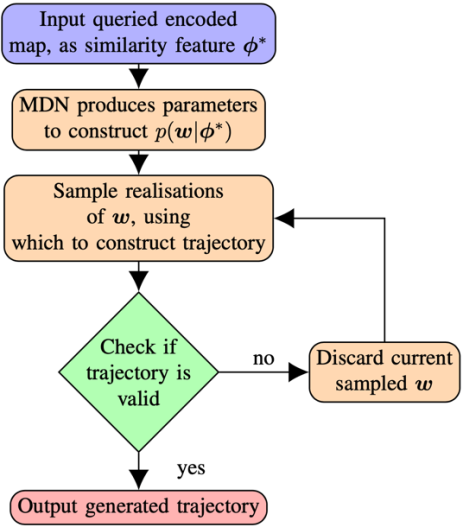


Рисунок 3 - архитектура модуля генератора траекторий

# Статическая модель OCTNet (15 стр)

## Данные

## Представление траекторий

### Дискретные

### Непрерывные

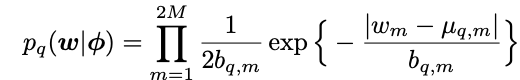
## Признаки модели

## Архитектура модели

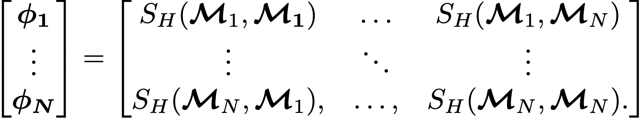
## Предсказания модели

## Генерация траектории по предсказанию

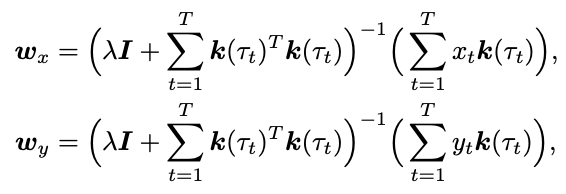
Как было казано ранее, OCTNet обучается на основе MDN сети. В качестве распределения выбрано распределение Лаплассса т.к. оно лучше отражает мультимодальный характер траекторий (1).

(1)

Нам необходимо обучить сеть вычислять параметры, получая на вход расстановку. После получения расстановки, проведем вычисление близости расстановки к уже известным нам - построим вектор свойств(2):

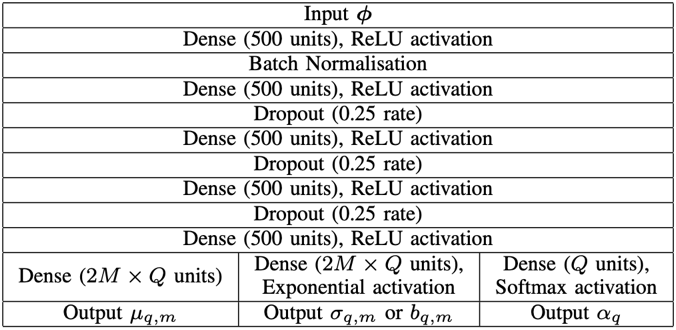
(2)

Также, необходимо закодировать траектории в вектора одинакового размера(3, 4):

(3, 4)

Затем, отформатированные данные пойдут на вход MDN, чтобы предсказать (1).

Архитектура MDN указана на рис. 4:



Риссунок 4 - архитектура MDN сети

# Динамическая модель OCTNet (20 стр)

## Данные

## Представление траекторий

### Дискретные

### Непрерывные

## Признаки модели

## Архитектура модели

## Предсказания модели

## Генерация траектории по предсказанию

Для построения динамической модели OCTNet можно использовать два способа: пошаговое построение с использованием статической модели на каждом шаге и LSTM сеть. Каждый способ имеет вои преимущества и недостатки.

Вариант с пошаговым построением алгоритма прост в реализации и работает достаточно быстро благодаря тому, что OCTNet представляет из себя непрерывную функцию, которая для каждого момента t [0..1] способна выдать точку маршрута. Данный способ был реализован, после чего был проведен замер его точности и сравнение с KTM. Для реализации данного подхода необходимо:

Обучить статическую модель OCTNet

Запустить модель из исходной точки

Выбрать траекторию и сместиться в нее

Запустить модель из новой точки

Отбросить не корректные или ломанные маршруты

Если таковых не имеется, вернуться в предыдущую точку и повторить попытку.

Продолжать до конца маршрута

Простота данного способа обеспечивает простоту его реализации и предсказуемость действий. Однако, данный подход заставляет делать дополнительные шаги и не использует данные о наблюдаемой ранее динамике.

LSTM - сеть с кратковременной памятью, помогает решить проблему с использованием предыдущих знаний о динамике системы, чтобы учесть это в будущих шагах. Данный момент в данный момент находится в разработке и не будет описан в рамках данной статьи.

# Сравнения точности работы алгоритмов (15 стр)

## Сравнение статических моделей

### GAN

### CVAE

### KTM

### OCTNet

## Сравнение динамических моделей

### GAN

### CVAE

### KTM

### OCTNet

Проведем сравнение точности работы алгоритмов. Точность будет измеряться следующим образом: для каждой корректно сгенерированной траектории будет вычислено расстояния Хаусдорфа и Фретчета до каждой предоставленной на вход траектории. Затем будет браться минимальное расстояние. Такой замер произведен для каждого алгоритма.

Расстояние Хаусдорфа вычисляется по формуле (5), а Фречета по формуле 6).

(5)

 (6)

Для статической среды произведем сравнение OCTNet с CVAE и GAN. Результаты в Табл. 2.

Таблица 2 - точность в статических средах

|  |  |  |
| --- | --- | --- |
|  | Hausdorf | Frechet |
| **OCTNet** | **1.86** | **2.00** |
| CVAE | 9.48 | 14.67 |
| GAN | 11.79 | 16.66 |

Для сравнения точности в динамической среде возьмем OCTNet и KTM. Результаты в Табл. 3.

Таблица 3 - точность в динамичеких средах

|  |  |  |
| --- | --- | --- |
|  | Hausdorf | Frechet |
| OCTNet | 2.11 | 3.38 |
| KTM | 6.64 | 11.20 |

# Оценка и защита результатов интеллектуальной деятельности (10 стр)

# Заключение (1 стр)

В рамках данной исследовательской работы была разработана статическая модель OCTNet, проверена ее применимость к решению динамических задач, а так же проведено сравнения точности работы разработанного алгоритма и алгоритмов-аналогов. OCTNet достаточно хорошо показал себя в решении задачи генерации траекторий в статической и динамической среде, о чем свидетельствует раздел V. В дальнейшем необходимо реализовать LSTM-OCTNet модель, для возможности использовать ранее полученные данные о расстановках, чтобы учитывать динамику.

# Литература

* Jianmin B, Dong C, Fang W, Houqiang L , Gang H CVAE-GAN: Fine-Grained Image Generation through Asymmetric Training // arXiv. 2018.
* Weiming Z, Tin L, Lionel O, Gilad F, Fabio R OCTNet: Trajectory Generation in New Environments from Past Experiences // arXiv. 2019.
* Nicolas V, Oleksandr Z, Theodoros T, Priya G, Zachary D, William S, Sven V, Andrew A, Albert C, Tensor Comprehensions: Framework-Agnostic High-Performance Machine Learning Abstractions // arXiv. 2018.
* T. Lai, W. Zhi, and F. Ramos, “Occ-traj120: Occupancy maps with associated trajectories,” CoRR, 2019.
* Николенко С, Кадурин А, Архангельская Е Глубокое обучение. Погружение в мир нейронных сетей. СПб.: Питер, 2018.
* Марсикано К., Стюарт К., Филлипс Б. Android. Программирование для профессионалов. Питер, 2017.
* Нишант Ш Машинное обучение и TensorFlow. СПб.: Питер, 2018.
* Гудфеллоу Я., Бенджио И., Курвилль А. Глубокое обучение. СПб.: ДМК Пресс, 2018.
* Фрвансуа Ш Глубокое обучение на Python. СПб.: Питер, 2018.

# Приложение

Ссылка на код: https://github.com/Criptonite/OCTNet