МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

ОТЧЕТ

по научно-исследовательской работе

Тема: Реализация и исследование алгоритма генерации траекторий в динамической среде на основе OCTNet

Студент гр. 5303 _____ Губа Д.А.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2020

ЗАДАНИЕ

НА НАУЧНО-ИССЛЕДОВАТЕЛЬСКУЮ РАБОТУ

Студент Губа Д.А.

Группа 5303

Тема работы Реализация и исследование алгоритма генерации траекторий в динамической среде на основе OCTNet

Исходные данные:

Реализовать алгоритм генерации траекторий в динамической среде, изучить характеристики алгоритма

Дата выдачи задания: 10.09.2020

Дата защиты НИР: 20.12.2020

Студент _____ Губа Д.А.

Преподаватель _____ Жангиров Т.Р.

Календарный план работы на весенний семестр

Утверждаю

Зав. кафедрой МО ЭВМ

_____ Кринкин К.В.

«___»_____2020 г.

Студент Губа Д.А. Группа 5303

Тема работы: Реализация и исследование алгоритма генерации траекторий в динамической среде на основе OCTNet

№ п/п	Наименование работ	Срок выполнения
1	Реализация OCTNet в статической среде	10.01 – 31.01
2	Реализация OCTNet в динамической среде	01.02 – 28.02
3	Разработка демонстрационного приложения	01.03 – 31.03
4	Демонстрация текущих результатов работы	20.05 - 22.05

Руководитель _____ Жангиров Т.Р.

Содержание

Результаты работы в осеннем семестре

В рамках работы в осеннем семестре было проведено проектирование архитектуры системы, выбор инструментов для реализации программных модулей, а также были выбраны методы тестирования работоспособности и эффективности алгоритма.

Архитектура системы

Архитектура программной реализации состоит из 4 модулей:

- Модуль предобработки данных
- Модуль обучения модели
- Модуль генератора траекторий
- UI модуль для отображения результатов

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:

0- 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:

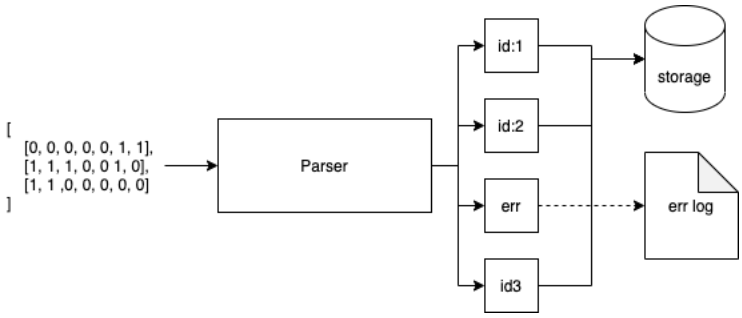

Рисунок 1 - потоковый парсер с обработкой отказов

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:

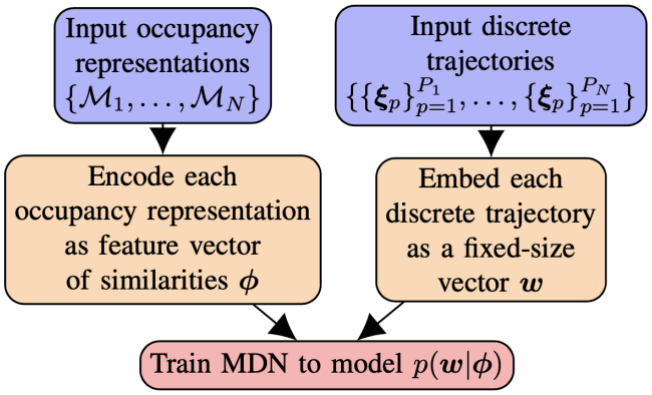

Рисунок 2 - архитектура модуля обучения модели

Модуль генератора траекторий

На вход 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:

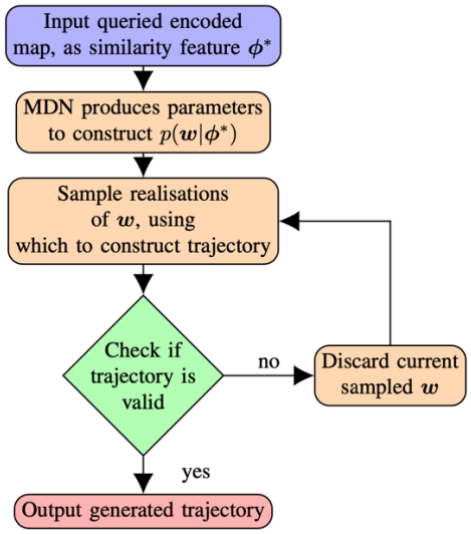

Рисунок 3 - архитектура модуля генератора траекторий

UI модуль для отображения результатов

Данный модуль будет реализован при помощи фреймворка ROS, а именно библиотеки rospy. Модуль будет отвечать за отображение данных в

репрезентативном для наблюдателя виде. Интерфейс для демонстрации изображен на рис. 4:

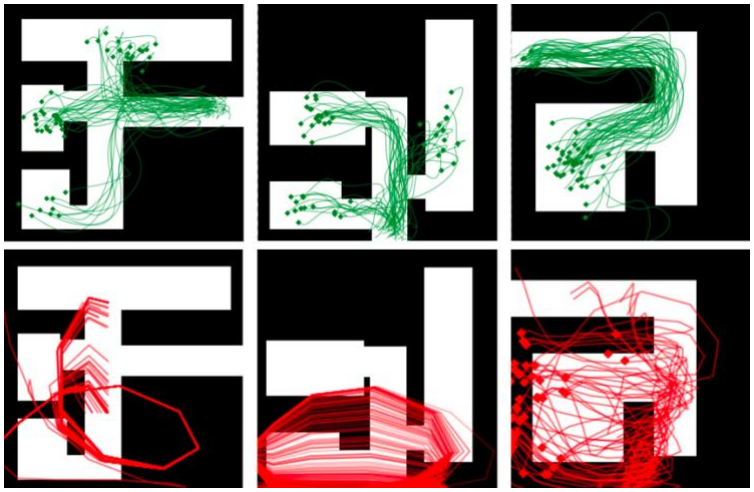

Рисунок 4 - интерфейс для демонстрации

Интерфейс будет отображать карту расстановки и построенные алгоритмом траектории. Корректные маршруты будут отмечены зеленым, а не корректные будут отмечены красным. При возможности будет реализован робот для модуля самоходного робота. На данный момент разработана архитектура модуля ROS, изображенная на рис 5:

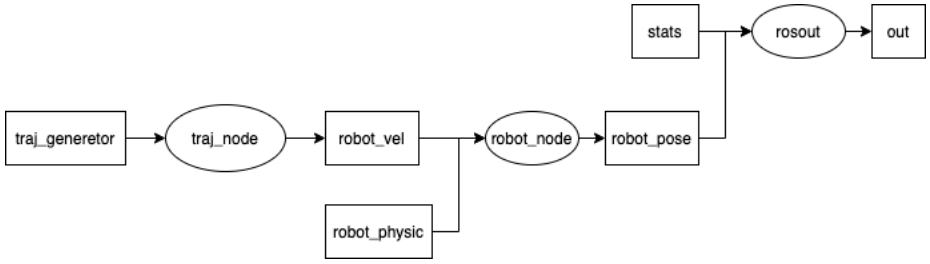

Рисунок 5 - архитектура модуля ROS

traj_generator - топик, который публикует координаты траектории движения

traj_node - узел, который подписывается на топик траекторий и передает данные в необходимом виде в канал движения робота

robot_vel - топик, который публикует движения для робота

robot_physics - внешний модуль физики для робота, который учитывает инерцию, ускорение, характер поверхности

robot_node - узел, который обрабатывает данные движения и физики и затем публикует следующее положение робота для отрисовки

robot_pose - топик, который получает положение робота и публикует их в канал выхода в необходимом виде

stats - побочный топик для получения структуры модулей

rosout - узел, отвечающий за вывод данных в интерфейс

out - топик для отображения графики

Инструменты для реализации системы

Для разработки программного решения будут использованы:

- Язык программирования python3
- Фреймворк для машинного обучения TensorFlow
- REST API для получения данных
- ROS для отображения результатов

Методы тестирования

Для подтверждения корректности алгоритма необходимо оценить его вероятностную модель. Вероятность построения не неверного маршрута планируется описать в виде цепей Маркова. Данный метод позволит вероятностно доказать успешность алгоритма. Для сравнения необходимо построить цепи Маркова для алгоритмов-конкурентов и сравнить с цепью Маркова для OCTNet.

Для проведения замеров будет использоваться метод определения среднего времени для построения траектории.

Заключение

В рамках работы за осенний семестр была выбрана архитектура приложения. Также, были изучены и выбраны инструменты для реализации работоспособности системы. Были выбраны методы для тестирования эффективности и работоспособности алгоритма.